Documentação Trabalho Prático 1 de AED's 2

Prof(a). Gisele Lobo Pappa

Aluno: Marcus Vinícius de Oliveira

1. Introdução

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 e capaz de computar a importância de paginas web de acordo com os links existentes entre as paginas. A intuição básica por trás desse algoritmo é que, se uma pagina é importante, então existe uma grande probabilidade de um usuário atingi-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 conexoes entre as paginas ´ como um grafo direcionado G(V, E), onde V consiste nas paginas (nós) é E consiste nos links entre paginas (arestas). Podemos representar tal grafo com uma matriz de adjacência de dimensão $|V| \times |V|$. Cada célula A_{ij} assume valor 1 se existe um link da i-ésima página para a j-ésima pagina, ou 0 caso contrário.

Para calcular as importâncias de cada página, o PageRank utiliza uma matriz S estocástica. Sem entrar em grandes detalhes, uma matriz e 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-esima página e chegar na j-ésima página em n passos calculando S^n . Considerando um numero 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, ate 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 pagina atual) e ´ e modelado no PageRank por meio de um fator conhecido como damping factor. Podemos adicionar essa informação na matriz estocástica descrita anteriormente utilizando a ´ formula ´ $M_{ij} = (1 - \alpha) * S_{ij} + \alpha * \frac{1}{N}$ (1) onde α e o damping factor e ´ N é o número de páginas.

2. Implementação

Funções e Procedimentos

O programa possui as seguintes funções:

double alocarMatriz(int Linhas,int Colunas):** Recebe dois inteiros 'Linhas' e 'Colunas' e aloca dinamicamente na memoria uma matriz do tipo double de tamanho 'Linhas' x 'Colunas'. Durante alocação, é atribuido o valor '0' a todos os elementos da matriz. Após alocar, retorna o endereço da matriz. Está função pode ser usada para alocar matrizes de qualquer dimensões.

void liberaMatriz(double **matriz, int dimensao): Recebe o endereço de memoria de uma matriz quadrada e sua dimensão e libera da memoria essa matriz. Está função só pode ser usada para liberar da memoria uma matriz quadrada.

void imprimeMatriz (double **matriz, int dimensao): Recebe o endereço de memoria de uma matriz quadrada e sua dimensão e a imprime na tela. Não é usada no programa final mas foi utilizada para testes durante a implementação. Está função só pode ser usada para imprimir uma matriz quadrada.

void substituiLinhas0 (double **matriz, int *vet_linhas, int dimensao): Recebe uma matriz de adjacência, sua dimensão e o vetor 'vet_linhas', construído durante a leitura de dados. O vetor 'vet_linhas' armazena valores booleanos. Se linha i da matriz possui ao menos um elemento diferente de 0, então vet_linhas=1, caso não possua, vet_linhas=0 . Caso 'vet_linhas[i]'==0, então linha matriz [i][j] é preenchida com 1's.

void matrizEstocastica (double **matriz, int dimensao): Recebe matriz de adjacência e sua dimensão e gera a matriz estocástica S da seguinte forma: Soma os elementos de uma matriz e utiliza o resultado como divisor na divisão de 1/soma_linha. Os elementos diferentes de 0 desta linha, são substituídos pelo resultado da divisão. O processo é repetido para todas as linhas. Desta forma fazemos com que cada linha de A some 1.

void gerarMatrizm (double **matriz_adjacencia, double **matriz_m, int dimensao, double damping_factor): Recebe matriz estocástica, uma matris_m que guardará o resultado da função, a dimensão das matrizes e o valor do fator damping. Cada elemento da matriz_m recebe como valor o resultado da equação:

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

void multiplicaMatriz (double **matriz_1, double **matriz_2, double **matriz_resultado, int dimensao): Recebe três matrizes e suas dimensões. A matriz_resultado recebe o valor da multiplicação da matriz_1 pela matriz_2.

void powMatriz (double **matriz_1, double **matriz_resultado, int dimensao, int expoente): Recebe matriz_1, matriz_resultado, a dimensão das matrizes e um número inteiro. A função faz a potenciação da matriz_1 pelo expoente recebido. Durante a execução, a função multiplicaMatriz é chamada 'expoente' vezes.

double somatorioMatriz (double **matriz, int dimensao): Recebe uma matriz e sua dimensão. Retorna o somatório dos elementos da matriz.

double subtraiMatriz (double **matriz_1, double **matriz_2, double **matriz_resultado, int dimensao):** Recebe três matrizes e suas dimensões. Subtrai matriz_2 de matriz_1 e guarda resultado em matriz_resultado.

void convergirMatriz (double **matriz_1, double **matriz_resultado, int dimensao): Recebe a matriz M como matriz_1, uma matriz_resultado, e suas dimensões. Multiplica matriz M sucessivas vezes até que haja convergência e guarda resultado na matriz_resultado. A matriz m é multiplicada sucessivas vezes conforme equação:

$$\|M^m - M^n\| \leq 10^{-12} \text{ para } m > n, \text{ onde:}$$

$$\|B\| = \sum_i \sum_j B_{ij}^2 \qquad (2)$$

Programa Principal

O programa principal começa lendo as entradas e alocando as matrizes que serão utilizadas no algoritmo do PageRank. Para isso as função 'alocarMatriz' é chamada algumas vezes. Durante a leitura dos dados, uma matriz de adjacência é criada, juntamente com um vetor (vet_linhas) que guarda quais linhas da matriz possuem elementos e quais não possuem.

Após a leitura e construção da matriz de adjacência, a função 'substituiLinhas0' é chamada e todos os elementos das linhas nulas da matriz de adjacência são substituídos por 1.

Após este passo, a função 'matrizEstocastica' é chamada fazendo com que cada linha da matriz de adjacência some 1. Em seguida é chamada a função 'gerarMatrizm' que é responsável por gerar a matriz M a partir da matriz de estocástica conforme equação (1) supracitada.

Com a matriz M pronta uma copia é criada(matriz_m2) e ambas são enviadas para a função 'convergirMatriz'. Através de varias multiplicações sucessivas e obedecendo a condição de parada descrita pela função (2), acontece a convergência da matriz.

Nesse ponto, cada coluna mede a importância da pagina. É impresso as diversas colunas da primeira linha da matriz convergência juntamente com seu identificador.

As matrizes dinamicamente alocadas que restaram, são liberadas, o arquivo fechado e o programa finalizado.

3. Análise de Complexidade

A análise de complexidade será feita em função da variável 'dimensao' que representa a dimensão das matrizes quadradas, bem como o numero de paginas que será analisada.

Função 'alocarMatriz': A função executa alguns comandos O(1) e depois entra em um loop que é executado n vezes. Dentro desse loop são feitos alguns comandos O(1) e entra em um novo loop que também é executado n vezes. Dentro do segundo loop é executado uma atribuição. Portanto temos $O(1)*n.O(1)*n.O(1)= O(n^2)$

Função 'liberaMatriz': A função executa alguns comandos O(1) e depois entra em um loop que é executado n vezes. Dentro do loop é feito um único comando O(1). Logo sua complexidade é O(n).

Função 'imprimeMatriz': A função só foi usada durante o desenvolvimento do programa, entretanto possui um loop dentro de outro, cada um sendo executado n vezes. Dentro do segundo loop é executado um comando O(1). Logo a complexidade é $O(n^2)$.

Função 'substituiLinhas0': é semelhante a função 'imprimeMatriz'. Constituída de um loop dentro do outro, cada um sendo executado n vezes. Dentro do segundo loop é executado um comando O(1). Logo a complexidade é $O(n^2)$.

Função 'matrizEstocastica': Possui um primeiro loop que é executado n vezes. Dentro deste loop existem mais dois, cada um executado n vezes. Dentro de cada um, atribuições e operações O(1). Logo temos que a complexidade é $O(n^2)$.

Função 'gerarMatrizm': Possui dois loops, um dentro do outro. Cada um sendo executado n vezes. Dentro do segundo loop alguns comandos da ordem O(1) são executados. Temos então que a complexidade é $O(n^2)$

Função 'multiplicaMatriz': Possui três loops, um dentro do outro sendo executados n vezes cada. Dentro do ultimo loop existem algumas operações O(1). Logo temos que a complexidade é $O(n^3)$.

Função 'powMatriz': A primeira parte da função é constituida de dois loops, um dentro do outro, executados n vezes cada. No ultimo loop existe uma operação O(1).

Na segunda parte , existe um loop que é executado n vezes. Dentro deste loop existe uma chamada para a função 'multiplicaMatriz' que é $O(n^3)$. Ainda dentro deste loop existem mais dois, um dentro do outro.No ultimo loop existem duas operações O(1).

Assim, temos que a complexidade é de: $O(n^2) + (O(n)*(O(n^3)+O(n^2) = O(n^4)$.

Função 'somatorioMatriz': Constituída de um loop dentro do outro, cada um sendo executado n vezes. Dentro do segundo loop é executado um comando O(1). Logo a complexidade é $O(n^2)$.

Função 'subtraiMatriz': Segue o mesmo padrão da 'somatorioMatriz'.É constituída de um loop dentro do outro, cada um sendo executado n vezes. Dentro do segundo loop é executado um comando O(1). Logo a complexidade é $O(n^2)$.

Função 'convergirMatriz': A primeira parte da função é constituída de dois loops, um dentro do outro, executados n vezes cada. No ultimo loop existem duas operações O(1).

Na segunda parte, existe um loop que é executado p vezes. Dentro deste loop existe duas chamadas para a função 'powMatriz' que é $O(n^4)$ e uma para 'subtraiMatriz' $O(n^2)$ e outra para a função 'somatorioMatriz' que é $O(n^2)$. Ainda dentro deste loop existem mais dois, um dentro do outro.No ultimo loop existem três operações O(1).

Logo temos que: $p\ (2O(n^4)+2O(n^2)+O(n^2)) = p\ O(n^4)$

Main: Apresenta vários comandos constantes e chama 3 vezes 'alocarMatriz' e o liberaMatriz, 1 vez os 'substituiLinhas0', 'matrizEstocastica', 'gerarMatrizm' e o 'convergirMatriz'. Além disso, é usado dois loops, um dentro do outro e um sozinho, cada um sendo executado n vezes.

Com isso, termos que a complexidade da main é: $3O(n^2)+3O(n)+O(n^2)+O(n^2)+O(n^2)+O(n^4) = O(n^4)$

4. Conclusão

Com exceção do caso teste 6, a implementação do trabalho transcorreu sem maiores problemas. Houve uma dificuldade inicial com relação a interpretação da formula (II), mas a dificuldade foi superada com desenvolver do problema.

Referências

Ziviani, N., Projeto de Algoritmos com Implementações em Pascal e C, 2ª Edição, Editora Thomson, 2004.

http://www.cplusplus.com/reference/cstdio

http://www.mspc.eng.br/info/cpp_oper_10.shtml

https://ssa_mat.catalao.ufg.br/up/615/o/matrizes_google.pdf

Slides da aula.

Anexos

Listagem dos programas:

• PageRank.h

• PageRank.c

• main.c