PageRank

Trabalho Prático 1 - AEDS II

Aluno: Luiz Otávio Resende Vasconcelos
Matrícula: 2015042142

- Introdução

Este visa descrever o desenvolvimento do algorítmo PageRank utilizado, por exemplo, pelo Google para ordenar as páginas de maior importância.
Verificamos a importância de cada página olhando para a quantidade de páginas que a referenciam, ou seja, quantas páginas contém um *link* para tal página.
Para tal, utilizaremos matrizes onde cada coluna representará uma página com as devidas probabilidades de chegarmos nela a partir de outra página.

- Implementação

Linguagem de programação utilizada: C

No início da implementação, recebemos 2 parâmetros na entrada:
- N = número de páginas
- df = Damping Factor
E após isso, *t-uplas* que representam em quais posições *i* e *j* valerá 1 na matriz.
(o resto será igual a zero).

Foram utilizadas 4 matrizes para fazer os cálculos:

matrix
matrix_aux
matrix_answ
matrix_nrm

Para cada, fora utilizado a função ***createMatrix(int size)*** para alocar size*size posições na memória para tal matriz, tendo essa função um retorno da referência das posições alocadas
sendo atribuída à matriz, fazendo assim, a matriz fazer uma referência à posição alocada.

Como especificado, agora se faz necessário verificar se a matriz tem alguma linha com apenas zeros. Para tal, utilizei a função

swapZeroLine(double** matriz, int size)

que recebe a matriz(por referência) e seu tamanho. Verifica-se então se há uma linha de apenas zeros e faz a troca por 1.

Após isso, precisamos obter a matriz estocástica. Onde a soma de cada linha é igual a 1.
Utilizei a função

getStochasticMatrix(double** matrix, int size)

que basicamente contará quantos 1's têm em cada linha, e dividirá cada um pelo total.

Com a matriz estocástica em mãos é aplicado o *Damping Factor* com a função

dampingFactor(double** matrix, double df, int size)

onde é passada a matriz por referência e é aplicada a fórmula do Damping Factor em cada elemento:

$$Mij = (1 - \alpha) * Sij + \alpha * 1/N$$

onde $\alpha = df$
N = size
Sij = matriz estocástica

Agora que temos a matriz com o *Damping Factor* aplicado, fazemos multiplicações sucessivas das matrizes e observar uma convergência.

Para multiplicar as matrizes, utilizei a função

pow2Matrix(double matrix, double** matrix_aux, double** matrix_answ, int size)**

onde **matrix** e **matrix_aux** são as matrizes a serem multiplicadas e **matrix_answ** será a matriz que receberá o resultado.

Então multiplicamos, recebemos o resultado na **matrix_answ,** atribuímos os valores de **matrix_answ** a **matrix_aux** e repetimos o processo.

Quando parar?

-> *while(n < 2000 && nrm > 0.00000000001)*

ou seja, quando **n** >= 2000, sendo um **n** iniciando em 0 e recebendo n+1 a cada loop.
e **nrm** sendo a norma da **(matriz seguinte - matriz anterior)** for <= 10^(-12)

A norma é calculada com a função

double norm(double **matrix,int size);

que basicamente recebe a matriz, eleva cada termo ao quadrado e soma em uma variável que é retorna ao final do loop.

Se o loop pára significa que chegamos onde queríamos:

-> Houve uma convergência na matriz.
Agora todas as linhas são iguais e a coluna de maior valor, representa a matriz de maior importância.

- Análise de complexidade

Temos que a função de maior complexidade assintótica é

pow2Matrix(double matrix, double** matrix_aux, double** matrix_answ, int size)**

que é $O(n^3)$.
Dentro dela, há algumas atribuições que são O(1).
Mas o quê domina assintóticamente são os 3 laços :

$O(f(n))O(g(n))O(h(n)) = O(f(n)g(n)h(n)) = O(n^3)$.

Porém, podemos afirmar que nosso algorítmo tem Complexidade Assintótica Firme O(n^4).
Pois essa função está dentro de um outro laço.

Analizemos as outras funções

double** createMatrix(int size)
que é $O(n)$. Apenas atribuições O(1) e um laço.

void swapZeroLine(double** matrix, int size) e
void getStochasticMatrix(double** matrix, int size)
que são $O(n^2)$. Mesmo considerando a pior das hipóteses, temos que
$O(f(n) + g(n)).O(h(n)) = O(n^2)$ em ambas

void dampingFactor(double** matrix, double df, int size)
double norm(double **matrix,int size)
que são $O(n^2)$.
temos apenas uma atribuição O(1) e dois laços em ambas.