

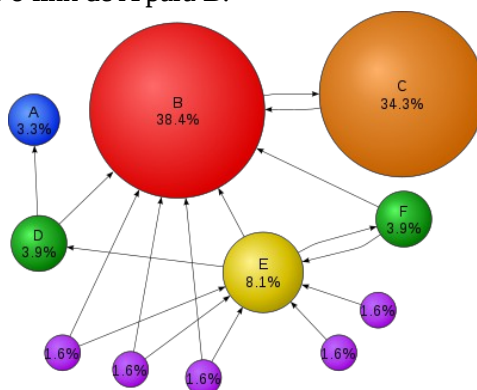
Documentação do Trabalho Prático de AEDS II

Julia Tiemi Alkmim Morishita

1.Introdução

Para qualquer pesquisa realizada no Google, existem milhares de páginas qualificáveis como úteis. A fim de determinar quais páginas devem aparecer no topo da pesquisa, foram desenvolvidos diversos algoritmos, sendo um deles o PageRank.

Criado por Larry Page, o PageRank determina o quão relevante é uma página, baseando-se na quantidade e na qualidade de links direcionados a um certo site. Observe a figura 01. Os nós representam as páginas e as arestas o link de A para B.



Para construir o PageRank, podemos utilizar esse grafo e construir uma matriz. Em cada elemento a_{ij} , será apresentado o valor 0 (false) ou 1 (true) para se a i -ésima página possui um link para a j -ésima página. Para calcular a probabilidade de sairmos de uma página para a outra, transformamos essa matriz numa matriz estocástica e fazemos seu quadrado até ela convergir.

Considerando também a probabilidade de a página ser acessada pelo acesso direto do URL, adicionamos um fator conhecido como Damping Factor.

2.Implementação

Foi implementada uma matriz alocada dinamicamente que recebe a informação sobre quais páginas tem links para quais páginas. Foram alocadas com a função `calloc`, para já possuir os elementos zerados onde as informações não faziam referência.

No próximo passo, foi necessário transformar linhas nulas em linhas apenas com elementos 1. Para isso, um contador foi passando de linha por linha. Uma vez que ele encontrava um elemento 0, somava-se +1 no contador. Se este se igualasse ao tamanho da linha, todos seus elementos eram substituídos.

Para transformar a matriz em uma matriz estocástica, cada elemento de uma linha foi dividido pela soma de todos esses elementos.

Após isso, cada elemento foi transformado pela equação do Damping Factor:

$$E_{ij} = (1 - DF) * A_{ij} + DF * 1 / N$$

Para chegar à resposta final, basta multiplicar a matriz até atingir sua conversão. Isso foi feito recursivamente, onde a condição de parada é a comparação da norma da matriz inicial subtraída da matriz ao quadrado com 10^{-12} :

$$\|M_m - M_n\| \leq 10^{-12}$$

3. Análise de Complexidade

****Allocate(int a):** $O(n) = n^2$

ZeroToOne(double *A, int b): $O(n) = n$

IsItZero(double **A, int a): $O(n) = n^2$

StochasticMatrixPartTwo(double *A, int b, int c): $O(n) = n$

StochasticMatrix(double **A, int a): $O(n) = n^2$

MatrixM(double **A, int a, double b): $O(n) = n^2$

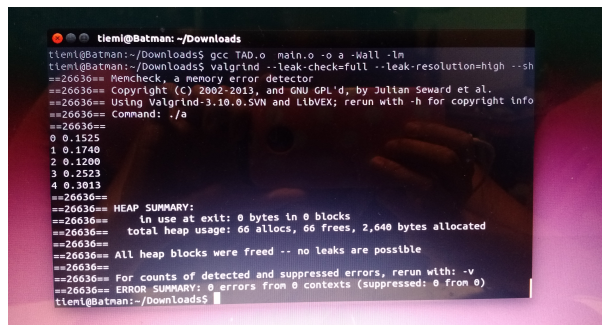
PrintAnswer(double **A, int a): $O(n) = n$

MatrixMultiplication(double **A, int a): $O(n) = n^3$

FreeMatrix(double **A, int a): $O(n) = n$

4. Testes

Segue a saída para o primeiro caso de teste.



```

tieni@Batman: ~/Downloads
tieni@Batman:~/Downloads$ gcc TAB.o main.o -o a -Wall -lm
tieni@Batman:~/Downloads$ valgrind --leak-check=full --leak-resolution=high --sh
==26636== Memcheck, a memory error detector
==26636== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==26636== Using Valgrind-3.10.0.SVN and LibVEX; rerun with -h for copyright info
==26636== Command: ./a
==26636==
0 0.1525
1 0.1740
2 0.1200
3 0.2523
4 0.3013
==26636==
==26636== HEAP SUMMARY:
==26636==    in use at exit: 0 bytes in 0 blocks
==26636==    total heap usage: 66 allocs, 66 frees, 2,640 bytes allocated
==26636==
==26636== All heap blocks were freed -- no leaks are possible
==26636==
==26636== For counts of detected and suppressed errors, rerun with: -v
==26636== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
tieni@Batman:~/Downloads$

```

5. Conclusão

A implementação do PageRank ocorreu sem muitos problemas e todos os casos de testes estiveram de acordo.

A principal dificuldade foi realizar a análise de complexidade da função recursiva. Não foi possível determinar com os conhecimentos adquiridos até agora.