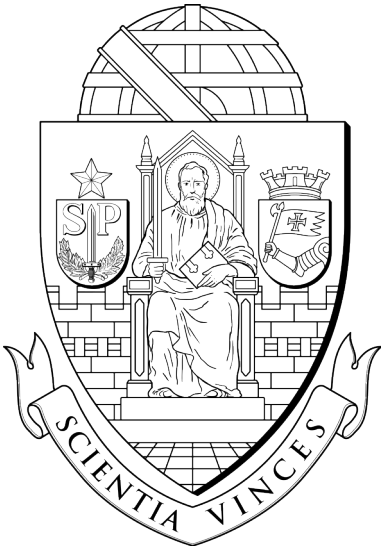

Universidade de São Paulo

Algorítmos e estrutura de dados 2

Mundo de Kevin Bacon

Alunos:
Henrique Gomes Zanin
Gabriel Guimarães Vilas Boas Marin

Sumário

1 Resumo

O presente trabalho propõe a exploração do mundo de Kevin Bacon. Para tanto utilizou-se um grafo não direcionado bipartido com o objetivo de disponibilizar a consulta do numero de Kevin Bacon de qualquer ator. O usuário também está habilitado a consultar a média do numero de Kevin Bacon para o mundo de Kevin Bacon bem como o desvio padrão.

2 Introdução

Um grafo pode ser definido como "Um conjunto de vértices e uma coleção de arestas com cada uma conectando um par de vértices"[Robert, 2011, p.518][2]. Há duas representações computacionais comuns para grafos: matrizes de adjacências e lista de adjacências, como exposto por Cormen(2002)[1].
A Matriz de adjacências é fundada sobre *arrays* ou vetores, estruturas disponíveis na maioria das linguagens de programação. Possuem a vantagem de serem de fácil implementação e complexidade 0(1) para acesso aos nós e arestas. Sua desvantagem reside no espaço necessário para acomodar os nós, sendo muita vezes n^2 [1].
A Lista de adjacências, por sua vez, não possui uma única estrutura de dado para apoiar-se, pode ser construída com: listas, para linguagens que as possuem como tipo primitivo; listas encadeadas; mapas, dicionários ou tabelas hash; sets ou conjuntos de valores únicos. Dessa forma a complexidade nas operações recuperação de arestas variam de acordo com a implementação. Para inserção de arestas, possuem 0(1) se as arestas forem adicionadas no início da lista, quando essa for encadeada. A estrutura de dado com o melhor desempenho é a tabela hash, que permite recuperar nós e arestas em 0(1).
Além das estruturas, grafos dependem de algoritmos de busca para operacionalizar o percurso que permite recuperar informações e caminhos. Há dois algoritmos que são base para a construção de outros algoritmos: busca em profundidade e busca em largura. A busca em profundidade é muito comum para percurso em árvore e quando aplicado a grafos gera uma arvore primeiro na profundidade, essa busca possui muita utilidade em inteligência artificial. Já a busca em largura gera um árvore primeiro em largura e, garante que o percurso gerado de um nó a outro é sempre o menor caminho[1], essa busca também origina diversos outros algoritmos como o Dijkstra para percurso em grafo ponderado, *topological sort*, entre outros.

[1] O espaço ocupado pode ser reduzido para **n** em grafos não direcionados

3 Implementação

Para construir o mundo de Kevin Bacon um grafo bipartido foi implementado com listas de adjacência.Todas as arestas são inseridas no início da lista, o que garante O(1) para inserção de nós e arestas. No grafo bipartido os nós filme estão conectados apenas a nós ator e ator apenas a filmes. Para o percurso no grafo foi utilizada a busca em largura para que o caminho de um ator para o Kevin Bacon seja o menor possível. A busca em largura é executada apenas uma vez, a partir dela um vetor de caminhos é gerado para que um caminho específico possa ser reconstruído. Dessa forma consultar o caminho de um ator para o Kevin Bacon exige em média 2,53 consultas ao vetor de caminhos[2].
Buscamos nesse trabalho usar diferentes níveis de abstração, onde cada *header* file evita que o usuário tenha que fazer chamadas de baixo nível para gerenciar estruturas fundamentais do grafo. Isso possibilitou a construção de um grafo genérico, sendo útil para solucionar diferentes problemas. A camada de mais baixo nível é a lista encadeada, utilizada pelo grafo. O *header file* que implementa o grafo garante que quem o utiliza não necessite fazer chamadas à lista encadeada. A ultima camada de abstração é o *sixDegree.h* que utiliza um dicionário para armazenar uma string e uma chave, evitando que o grafo tenha que manipular ponteiros genéricos. Essa estrutura também reduz a complexidade da consulta a um ator. Sua organização interna é um vetor ordenado lexicograficamente pela string, permitindo uma busca binária para recuperar o índice do nó. A inserção no dicionário é no pior caso linear e a busca O(Log(n)). Todo esse conjunto compõe a solução criada para administrar o mundo de Kevin Bacon. O número de Kevin Bacon, kb, de um ator é dado por:

$$kb = \frac{n-1}{2} \tag{1}$$

Com n representando o tamanho do vetor de caminho de um ator para o Kevin Bacon. O divisor consiste na quantidade de elementos do grafo bipartido, nesse caso filme e ator, totalizando 2 elementos. O caminho exibido muitas vezes não é único e está totalmente dependente da ordem de inserção na lista encadeada. Caso queira verificar outro caminho possível mude no graph.h em addDirectedEdge na linha 82 para addLast ao invés de addFirst. Atente-se para o fato de que isso aumentará a complexidade.

4 Operacionalização do grafo

Nesse trabalho, a maneira para operar o grafo é bem simples, usando apenas o básico do C e por isso sendo bem intuitivo, portável e eficaz.

[2]Valor extraído da consulta "Média aritmética e desvio padrão"

4.1 Comandos

Foram implementados os comandos existentes na descrição do trabalho. Logo, são encontrados as ordens, "inicializar", "pesquisar número de Kevin Bacon de um ator"e "Consultar média e desvio padrão dos atores".

4.1.1 Inicializar

Este comando, abre o arquivo "input-top-grossing.txt"com os dados e carrega tudo para o grafo. Após isso, a inicialização exerce sua busca em largura de todos os atores até o Kevin Bacon. Em seguida, calcula a média aritmética e o desvio padrão do número de Kevin Bacon dos atores.

Número do comando: 1

4.1.2 Pesquisar número Kevin Bacon de ator

Ao digitar o comando, o usuário deve inserir um nome do autor com o mesmo padrão que foi inserido. Assim que a informação for coletada, o programa busca a rota registrada do ator até o Kevin Bacon.

Número do comando: 2

4.1.3 Consultar média aritmética e desvio padrão

Executando essa instrução, o programa simplesmente acessa as duas variáveis que guardam a média aritmética e o desvio padrão calculadas na inicialização.

Número do comando: 3

4.1.4 Finalizar programa

Para sair do programa, esse comando da o "free"em todas as estruturas alocadas na memória, encerrando de uma maneira segura.

Número do comando: 0

Referências

[1] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Algoritmos: teoria e prática. *Editora Campus*, 2:296, 2002.

[2] Robert Sedgewick and Kevin Wayne. *Algorithms*. Addison-wesley professional, 2011.