Universidade Federal de São João Del-Rei

Grafos

Caracterização de Grafos

Wagner Lancetti 182050040

23 de Setembro de 2020

1 Apresentação

Grafos são ferramentas representativas que podem ser utilizados para resolverem diversos problemas reais. Alguns desses exemplos podem ser: construção de uma cidade, utilizando representação de bairros ou ruas; representrar comunicação entre redes sociais; e diversos outros exemplos em que haja conectividade entre as variáveis do problema. Desses exemplos apresentados podemos ter diversas aplicabilidades como, por exemplo, a interconectivade entre os bairros da cidade, bem como definir percursos que gaste menos tempo de ser percorrido; ou qual o assunto mais comentado dentro de uma rede social, analisando as partes em que o grafo é mais denso.

Sendo assim, criar um algorítmo que consiga fazer manipulação de grafos é fundamental para resolução de diversos problemas reais. O algoritmo apresentado a seguir tem como finalidade a manipulação simples de um grafo. As funcionalidades do algoritmo consistem em criação do grafo a partir de um arquivo de entrada informando as arestas (conexões) que existem entre os vértices, e com o grafo construído informar algumas carectrísticas interna dele, como a quantidade de vértices que ele possue, o grau de cada vértice, quem vem antes e quem vem depois de dum vértice específico, além de tratar da densidade do grafo.

2 Implementação

Para programar esse algoritmo foi utilizado linguagem C com representação matricial do grafo.

Inicialmente o programa abre um arquivo de entrada "graph.txt"que contêm 3 valores, sendo eles: vértice origem, vértice destino, peso da aresta. Segue um exemplo de entrada.

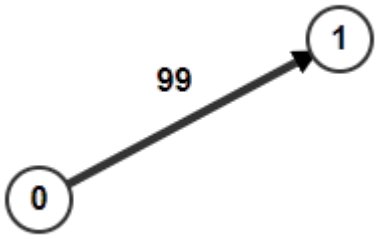

Figura 1: 0 1 99

Para montar a matriz do grafo foi necessário percorrer o arquivo de entrada duas vezes. A primeira varredura é para descobrir qual o tamanho deve ser reservado para a matriz, esse tamanho é definido pelo maior vértice, ou seja, se o maior vértice for 7, então é necessário reservar um espaço de memória para uma matriz 7x7; já a segunda varredura é para preencher a matriz com os valores informados no arquivo.

As funções foram divididas em 3 grupos, o primeiro para construção do grafo, o segundo para implementação das funções que buscam informações dentro do grafo e, por fim, a função principal que buscará o que for requerido. Abaixo serão aprensentadas as implementações de cada uma dessas funções.

2.1 Arquivo

Aqui serão apresentadas as funções que criam o grafo e manipulam o arquivo de entrada.

2.1.1 FindHigherVertex

Essa função é executada para encontrar o maior vértice dentro do grafo, ela faz uma comparação entre os dois vértices lidos na linha do arquivo com um vértice que é considerado o maior lido até o momento da interação. Após realizar todas as comparações ela retorna para o programa principal quanto deve ser alocado de espaço para a modelagem do grafo.

2.1.2 makeGraph

Essa função é tem como objetivo chamar as outras duas funções abaixo para montar todo o grafo. Ela recebe um grafo totalmente vazio e retorna para o programa principal o grafo montado a partir do dados lidos no arquivo de entrada.

2.1.3 makeMatrix

Essa função tem como único objetivo alocar espaço para a matriz que será o grafo. Ela recebe como entrada um ponteiro para uma matriz e retorna para a função makeGraph uma matriz alocada, com o tamanho encontrado na função FindHigherVertex e com todos os campos zerados.

2.1.4 CompleteGraph

Após ter a matriz alocada, essa função tem a finalidade de ler novamente o arquivo de entrada e preencher a matriz com os dados lidos, atribuindo os valores às ligações entre os vértices.

2.2 Atividade

Aqui serão apresentadas as funções que buscam carecterísticas dentro do grafo após ele ser construído.

2.2.1 Summary

Essa função recebe por referência os valores que ela precisa calcular, são eles:

- nodes: receberá o valor da quantidade de vértices que existem no grafo;
- edge: receberá o valor da quantidade de arestas que existem no grafo;
- density: receberá o valor do calculo da densidade do grafo.

Para calcular esses valores é necessário percorrer toda a matriz uma vez, para isso é travado a linha da matriz e são percorridas as colunas da matriz.

Como o programa em uma mesma interação verifica a presença de arestas e vértices, para calcular a quantidade de vértice foi necessário a criação de uma variável de controle "seeNodes", que basicamente verifica se a coluna que está sendo verificada é um vértice presente no grafo. Caso um valor $\neq 0$ seja encontrado, essa variável de controle altera para 'T' e aquele vértice é adicionado na contagem.

Enquanto realiza a operação descrita a cima, em cada interação nos campos da matriz é verificado se há arestas naquele ponto, caso encontre um valor $\neq 0$ essa aresta entra na contagem. Como um valor ≤ 0 não é válido para um peso de aresta, qualquer valor diferente disso indica para o programa que aquela aresta é válida.

Sabendo que a densidade do grafo é a razão entre a quantidade de arestas do grafo e a quantidade de arestas do grafo completo com o mesma quantidade de vértices, a fórmula $\left(\frac{2A}{V*(V-1)}\right)$ foi utilizada para o cálculo, sendo A a quantidade de arestas e V

a quantidade de vértices. Essa fórmula retorna um número maior que zero e menor que 1 ($0 \leq$ numero ≤ 1).

2.2.2 VertexDegree

Essa função tem a finalidade de encontrar o grau do vértice, para isso ela recebe o vértice que deseja ser verificado e o grafo. Sabendo que grau do vértice é a quantidade de arestas que chegam no vértice e que saem do vértice, foi necessário verificar a linha e coluna do vértice desejado, caso encontrado uma aresta, o grau do vértice era incrementado.

Além disso, atendendo a peculiaridade de self-loop para grau do vértice, que é contabilizado duas vezes, o programa resolve esse problema fazendo uma verificação específica no campo Grafo[vértice][vértice], caso haja uma aresta ali ela é contabilizada novamente.

2.2.3 Successors

Essa função tem a finalidade de encontrar todos os sucessores do vértice, como também a quantidade deles. As operações que essa função realiza é, a partir do vértice que deseja-se verificar, percorrer toda sua linha encontrando quais vértices ele possue arestas ligando.

Essa função teve que percorrer somente a linha do vértice devido à montagem do grafo, que foi feita pelo arquivo de entrada. Como descrito na seção 2.0 de Implementação, o arquivo de entrada possui os valores "Origem Destino Peso", e a montagem do grafo colocou os vértices de destino na linha daquele vértice específico.

2.2.4 Predecessor

Essa função tem a finalidade de encontrar todos os vértices que incidem no vértice que deseja-se verificar. Análogo ao que é feito na função 2.2.3, nessa função a coluna que o vértice pertente que é travada e a linha dele que é percorrida. O motivo disso foi a montagem do grafo, como explicado em 2.2.3, sendo que nas linhas do vértices ficaram as origens.

2.3 Main

Aqui será apresentada a função principal do programa, que controla o que o usuário deseja fazer.

2.3.1 main

A função principal do programa tem a finalidade de controlar as decisões do usuário, bem como de chamadas das funções de montagem do grafo. Essa função também se comunica com o usuário através de perguntas para decidir o quais funções da seção 2.2 serão chamadas.