TRABALHO DE ALGORITMOS EM GRAFO

O objetivo deste trabalho é projetar, desenvolver e validar uma biblioteca para manipular grafos. A biblioteca deve ser capaz de representar grafos, assim como implementar um conjunto de algoritmos em grafos. A implementação dessa biblioteca pode ser por meio de um conjunto de funções (em C, por exemplo) ou uma classe em uma linguagem orientada a objetos (em C++ ou Java por exemplo). Você deve projetar e desenvolver sua biblioteca de forma que ela possa ser facilmente utilizada em outros programas.

Para entregar o trabalho, o grupo irá preparar um relatório informando suas decisões de projeto e implementação das funcionalidades. Este relatório deve ter no máximo 5 páginas e será entregue juntamente com o código fonte da biblioteca e do programa utilizado para realizar o estudo de caso.

Descrição da primeira parte do trabalho

Seguem abaixo as funcionalidades que precisam ser oferecidas pela biblioteca na primeira parte do trabalho. Sua biblioteca irá trabalhar apenas com grafos não-direcionados e arestas sem peso, como mostrado na figura 1 ou com grafos não-direcionados e arestas com peso, como mostrado na figura 2.

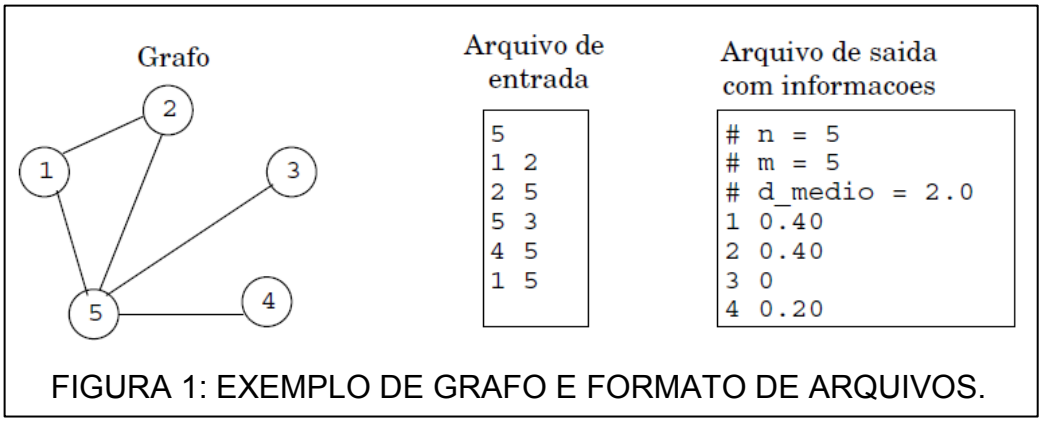

FIGURA 1: EXEMPLO DE GRAFO E FORMATO DE ARQUIVOS.

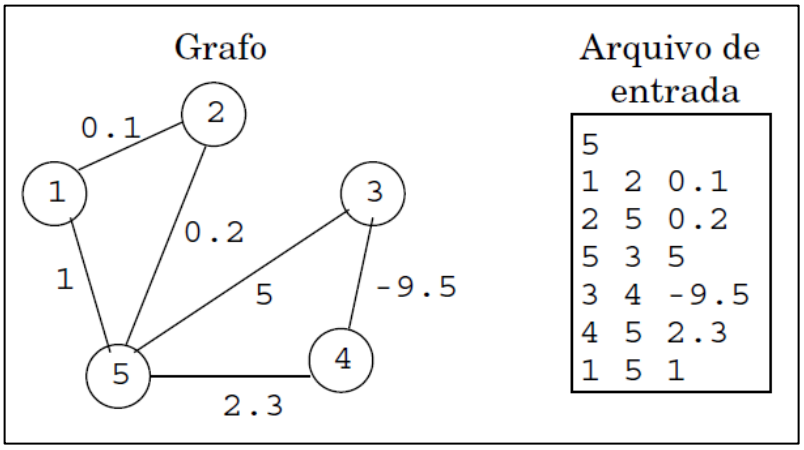

FIGURA 2: EXEMPLO DE GRAFO E FORMATO DE ARQUIVOS

O arquivo de entrada será único e terá a forma como mostrado na figura 2. Para os casos onde os pesos não são necessários, será possível ignorar a informação da terceira coluna.

1. **Entrada.** A biblioteca deve ser capaz de ler um grafo a partir de um arquivo texto. O formato do grafo no arquivo será o seguinte. A primeira linha informa o número de vértices do grafo. Cada linha subsequente informa as arestas. Um exemplo de um grafo e seu respectivo arquivo texto é dado na figura 1.
2. **Saída.** Sua biblioteca deve ser capaz de gerar um arquivo texto com as seguintes informações sobre o grafo: número de vértices, número de arestas e grau médio, e distribuição empírica do grau dos vértices. A Figura 1 ilustra o formato deste arquivo de saída para o grafo correspondente.
3. **Representação de grafos.** Sua biblioteca deve ser capaz de representar grafos utilizando tanto uma matriz de adjacência, quanto uma lista de adjacência. O usuário da biblioteca (programa que irá usá-la) poderá escolher a representação a ser utilizada.
4. **Busca em grafos: largura e profundidade.** Sua biblioteca deve ser capaz de percorrer o grafo utilizando busca em largura e busca em profundidade. O vértice inicial será dado pelo usuário da biblioteca. A respectiva árvore de busca deve ser gerada assim como o nível de cada vértice na árvore (nível da raiz é zero). Estas informações devem ser impressas em um arquivo. Para descrever a árvore gerada, basta informar o pai de cada vértice e seu nível no arquivo de saída.
5. **Componentes conexos.** Sua biblioteca deve ser capaz descobrir os componentes conexos de um grafo. O número de componentes conexas, assim como o tamanho (em vértices) de cada componente e a lista de vértices pertencentes à componente. Os componentes devem estar listados em ordem decrescente de tamanho (listar primeiro

o componente com o maior número de vértices, etc).

Descrição da segunda parte do trabalho

Esta é a segunda parte do trabalho prático da disciplina. Você deve realizar esta parte utilizando a biblioteca implementada na primeira parte. Se você fez a primeira parte em grupo, então o grupo deve continuar o mesmo. Para entregar o trabalho, prepare um relatório informando suas decisões de projeto e implementação das funcionalidades. Este relatório deve ter no máximo 5 páginas e será entregue juntamente com o código fonte da biblioteca e do programa utilizado para realizar os estudos de caso.

O que segue são as funcionalidades que precisam ser implementadas pela sua biblioteca de grafos para a segunda parte do trabalho.

1. Grafos com pesos. Sua biblioteca deve ser capaz de representar e manipular grafos não-direcionados que possuam pesos nas arestas. Os pesos, que serão representados por valores reais, devem estar associados às arestas. Você deve decidir a melhor forma de estender sua biblioteca de forma a implementar esta nova funcionalidade. O arquivo de entrada será modificado, tendo agora uma terceira coluna, que representa o peso da aresta (podendo ser qualquer número de ponto flutuante). Um exemplo de grafo não-direcionado com pesos e seu respectivo arquivo de entrada está ilustrado na figura acima.
2. Distância e caminho mínimo. Sua biblioteca deve ser capaz de encontrar a distância entre qualquer par de vértices assim como o caminho que possui esta distância. Se o grafo não possuir pesos, o algoritmo de busca em largura deve ser utilizado. Se o grafo possuir pesos, o algoritmo de Dijkstra deve ser utilizado. Neste último caso, é necessário verificar se os pesos de todas as arestas são maiores ou iguais a zero, condição necessária para que o algoritmo de Dijkstra funcione corretamente. Você deve decidir como implementar o algoritmo de Dijkstra em sua biblioteca (por exemplo, usando um heap), lembrando que isto irá influenciar o tempo de execução do seu algoritmo. Além de calcular a distância e caminho mínimo entre um par de vértices, sua biblioteca deve ser capaz de calcular a distância e caminho mínimo entre um dado vértice e todos os outros vértices do grafo.