**Implementação**

**Atividade em Grupo do Projeto**

|  |  |
| --- | --- |
| Nome do Integrante (ordem alfabética) | RA |
|  |  |
|  |  |
|  |  |

Tendo por base a linguagem de programação a ser utilizada no desenvolvimento do seu projeto da disciplina, faça os exercícios a seguir.

**Matriz de Adjacência – Grafos Direcionados e Não direcionados**

1. Escreva um método “int inDegree(int v)” que calcula e retorna o grau de entrada de um vértice **v** de um grafo dirigido. O método deve ser implementado na classe TGrafo da matriz de adjacência. Obs.: Grau de entrada de v é o total de arestas que chegam no vértice v.
2. Escreva o método outDegree(int v) que calcula o grau de saída de **v** em grafo dirigido. O método deve ser implementado na classe TGrafo que usa matriz de adjacência. Obs.: Grau de saída de v é o total de arestas que saem do vértice v.
3. Escreva um método para um grafo direcionado que recebe um vértice como parâmetro e retorne 1 se vértice for uma **fonte** (grau de saída maior que zero e grau de entrada igual a 0), ou 0 caso contrário. O método deve ser implementado para a classe TGrafo como matriz de adjacência.
4. Escreva um método para um grafo direcionado que recebe um vértice como parâmetro, retorne 1 se vértice for um **sorvedouro** (grau de entrada maior que zero e grau de saída igual a 0), ou 0 caso contrário. O método deve ser implementado para a classe TGrafo que utiliza matriz de adjacência.
5. Escreva um método que receba um **grafo dirigido** como parâmetro e **retorna 1** se o grafo for **simétrico** e 0 caso contrário. O método deve ser implementado para a classe TGrafo que utiliza matriz de adjacência.
6. Um **grafo** pode ser armazenado em um arquivo com o seguinte formato:

6

8

0 1

0 5

1 0

1 5

2 4

3 1

4 3

3 5

Onde na primeira linha contém um inteiro **V** (vértice), na segunda contém um inteiro **A** (arestas) e nas demais linha contém dois inteiros pertencentes ao intervalo 0..V-1. Se interpretarmos cada linha do arquivo como uma **aresta**, podemos dizer que o arquivo define um **grafo** com vértices 0..V-1. Escreva um método que receba um nome de arquivo com o formato acima e construa a representação do grafo como matriz de adjacência. (Exemplo de código que manipula arquivo no próximo slide)

1. Criar uma outra classe TGrafoND e modifique as funções insereA, removeA e show para representar um **grafo não-dirigido** utilizando matriz de adjacência.
2. Modifique a classe TGrafoND e os métodos correspondentes para permitir a criação de um grafo não direcionado rotulado (valor float) nas arestas.
3. Fazer um método que permita remover um vértice do Grafo (não dirigido e dirigido). Não se esqueça de remover as arestas associadas.
4. Fazer um método que verifique e retorne se o grafo (não dirigido) é completo.
5. Fazer um método que verifique e retorne e o grafo (dirigido) é completo.
6. Fazer um método que retorne o complemento (grafo complementar) de um grafo (dirigido ou não) na forma de uma matriz de adjacência.
7. Fazer um método que retorne o tipo de conexidade de um grafo não direcionado (0 - conexo ou 1 - não conexo – desconexo).
8. Fazer um método que retorne a categoria de conexidade para um grafo direcionado (3 – C3, 2 – C2, 1 – C1 ou 0 – c0).
9. Faze um método que retorne o grafo reduzido de um grafo direcionado no formato de uma matriz de adjacência.
10. Modifique a classe TGrafo e os métodos correspondentes para permitir a criação de um grafo direcionado rotulado (valor float) nas arestas.

**Lista de Adjacência – Grafos Direcionados e Não direcionados**

1. Escreva um método que decida se dois grafos direcionados são iguais. O método deve ser implementado para a classe TGrafo faz uso da lista de adjacência.
2. Escreva um método que converta uma representação de um grafo em outra. Por exemplo, converta um grafo armazenado em matriz de adjacência em uma lista de adjacência.
3. Escreva um método que receba um grafo armazenado em **lista de adjacência** e inverta a lista de adjacência de todos os vértices do grafo.  Por exemplo, se os 4 vizinhos de um certo vértice **u** aparecem na lista **adj[u]**na ordem **v**, **w**, **x**, **y**, então depois da aplicação do método a lista deve conter os mesmos vértices na ordem **y**, **x**, **w**, **v**. Obs.: Vizinhos são todos os vértices ligados ao vértice u.
4. Escreva um método que receba um grafo e um vértice como parâmetro e retorne 1 se vértice for uma **fonte** (grau de saída maior que zero e grau de entrada igual a 0), ou 0 caso contrário. O método deve ser implementado para a classe TGrafo como lista de adjacência.
5. Escreva um método que receba um grafo e um vértice como parâmetro, retorne 1 se vértice for um **sorvedouro** (grau de entrada maior que zero e grau de saída igual a 0), ou 0 caso contrário. O método deve ser implementado para a classe TGrafo que utiliza lista de adjacência.
6. Escreva um método que receba um **grafo dirigido** como parâmetro e **retorna 1** se o grafo for **simétrico** e 0 caso contrário. O método deve ser implementado para a classe TGrafo que utiliza lista de adjacência.
7. Um **grafo** pode ser armazenado em um arquivo com o seguinte formato:

6

8

0 1

0 5

1 0

1 5

2 4

3 1

4 3

3 5

Onde na primeira linha contém um inteiro **V** (vértice), na segunda contém um inteiro **A** (arestas) e nas demais linha contém dois inteiros pertencentes ao intervalo 0..V-1. Se interpretarmos cada linha do arquivo como uma **aresta**, podemos dizer que o arquivo define um **grafo** com vértices 0..V-1. Escreva um método que receba um nome de arquivo com o formato acima e construa a representação de lista de adjacência do grafo.

1. Fazer um método que permita remover um vértice do Grafo (não dirigido). Não se esqueça de remover as arestas associadas.
2. Fazer um método que permita remover um vértice do Grafo (dirigido). Não se esqueça de remover as arestas associadas.
3. Fazer um método que verifique se o grafo (dirigido ou não) é completo.

**O relatório a seguir deve conter o código fonte de cada um dos exercícios relacionados a implementação de Grafos representados como Matriz e Lista de Adjacência, incluindo um teste de cada um deles (printscreen).**

Ao enviar pelo Ambiente Virtual, não se esquecer de incluir o relatório e os códigos fontes de cada um deles na linguagem de programação utilizada para implementação (C++, C, Java ou Python) em um único arquivo compactado.

**Obs.: Esta atividade tem o peso equivalente a 4 (quatro) atividades. A nota do grupo será proporcional a quantidade de exercícios implementados de forma correta.**

**Relatório**