Trabalho 3 de Teoria dos Grafos

1.0

Generated by Doxygen 1.9.8

Chapter 1

Trabalho de Grafos

1.1 Membros da Equipe

- Lukas Freitas de Carvalho - Matrícula: 202376033

1.2 Descrição

Este repositório contém a implementação de um trabalho de Teoria dos Grafos. O projeto inclui a criação e manipulação de grafos utilizando listas encadeadas e matrizes de adjacência, com funcionalidades avançadas como cálculo de distâncias utilizando o algoritmo de Dijkstra.

1.3 Funcionalidades Principais

1.3.1 Grafo

- `getVertice(int id)`: Retorna o nó caso exista o id desejado.
- `getAresta(int origem, int destino)`: Retorna a aresta caso exista.
- `insereVertice(float val)`: Insere um vértice com peso.
- `insereAresta(int origem, int destino, float val)`: Insere uma aresta entre dois nós com peso.
- `removeVertice(int id)`: Remove o vértice com o id passado como parâmetro.
- `removeAresta(int origem, int destino)`: Remove a aresta caso exista.
- `eh_direcionado()`: Retorna se o grafo é direcionado ou não.
- `verticePonderado()`: Retorna se os vértices do grafo são ponderados ou não.
- `arestaPonderada()`: Retorna se as arestas do grafo são ponderadas ou não.
- `getOrdem()`: Retorna a ordem do grafo.
- `setOrdem(int val)`: Define a ordem do grafo.

- `setDirecionado(bool val)`: Define se o grafo é direcionado ou não.
- `setVerticePonderado(bool val)`: Define se o grafo possui vértices ponderados.
- `setArestaPonderada(bool val)`: Define se o grafo possui arestas ponderadas.
- `getGrau()`: Retorna o grau do grafo.
- `eh_completo()`: Retorna se o grafo é completo.
- `carregaGrafo(string grafo)`: Carrega o grafo na estrutura.
- `imprimeGrafo()`: Imprime o grafo de acordo com a descrição do trabalho.
- `maiorMenorDistancia(int ponto1, int ponto2)`: Calcula a menor distância entre dois pontos utilizando o algoritmo de Dijkstra.
- `retornaMenorDistancia()`: Calcula dentre todas as menores distâncias no grafo, a maior delas.
- `coloracaoArestaGuloso()`: Retorna a menor quantidade de cores nas arestas utilizando um algoritmo guloso.
- `coloracaoArestaRandomizado()`: Retorna a menor quantidade de cores nas arestas utilizando um algoritmo randomizado.
- `coloracaoArestaReativo()`: Retorna a menor quantidade de cores nas arestas utilizando um algoritmo reativo.

1.3.2 Grafo_lista

- `insereVertice(float val)`: Insere um vértice com peso.
- `insereAresta(int origem, int destino, float val)`: Insere uma aresta entre dois nós com peso.
- `removeAresta(int i, int j)`: Remove uma aresta entre dois nós.
- `removeVertice(int id)`: Remove um vértice.
- `getVertice(int id)`: Retorna o nó caso exista o id desejado.
- `getAresta(int origem, int destino)`: Retorna a aresta caso exista.

1.3.3 Grafo_matriz

- `insereVertice(float val)`: Insere um vértice com peso.
- `insereAresta(int origem, int destino, float val)`: Insere uma aresta entre dois nós com peso.
- `removeAresta(int origem, int destino)`: Remove uma aresta entre dois nós.
- `removeVertice(int id)`: Remove um vértice.
- `getVertice(int id)`: Retorna o nó caso exista o id desejado.
- `getAresta(int origem, int destino)`: Retorna a aresta caso exista.

1.3.4 Node

- `getProx()`: Retorna o próximo nó.
- `setProx(Node prox)`: Define o próximo nó.
- `setValue(float value)`: Define o valor do nó.
- `getValue()`: Retorna o valor do nó.
- `setId(int val)`: Define o id do nó.
- `getId()`: Retorna o id do nó.

1.3.5 NodeEdge

- `getPeso()`: Retorna o peso da aresta.
- `setPeso(float val)`: Define o peso da aresta.

1.3.6 NodeVertex

- `getArestas()`: Retorna a lista de arestas do vértice.
- `getGrau()`: Retorna o grau do vértice.
- `setGrau(int val)`: Define o grau do vértice.

1.3.7 Linked_list

Classe template para criação do grafo por lista encadeada.

- `getTam()`: Retorna o tamanho da lista.
- `getNodeById(int val)`: Retorna o nó correspondente ao id passado.
- `getUltimo()`: Retorna o último nó.
- `getPrimeiro()`: Retorna o primeiro nó.
- `insereFinal(float val)`: Insere nó no final da lista.
- `removeNode(NodeType no)`: Remove o nó passado como parâmetro da lista.

1.3.8 Linked_Vertex

Implementação da classe **`Linked_list`** (p. **??**) para vértices.

- `insereAresta(int origem, int destino, float val)`: Insere uma aresta entre dois vértices.
- `removeAresta(int i, int j)`: Remove uma aresta entre dois vértices.
- `removeVertice(int id)`: Remove um vértice.

1.4 Instalação

Para clonar e executar este projeto, você precisará do Git e de um compilador C++ instalado em seu sistema. No seu terminal:

```
# Clone o repositório
git clone https://github.com/Lukas712/Trabalho_de_grafos.git

# Entre no diretório do projeto
cd Trabalho_de_grafos

# Compile os arquivos
g++ -o {nome do programa} main.cpp src/*.cpp -I/.include/ -g -Wall -Werror
```

1.4.1 Uso

Após a instalação, você pode executar o programa utilizando os comandos abaixo:

```
./{nome do programa} -d -l {nome do arquivo contendo o grafo}.txt
```

ou

```
./{nome do programa} -d -m {nome do arquivo contendo o grafo}.txt
```

ou

```
./{nome do programa} -p -l {nome do arquivo contendo o grafo}.txt
```

ou

```
./{nome do programa} -p -m {nome do arquivo contendo o grafo}.txt
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Grafo Class Reference

Inheritance diagram for Grafo:

5.2 Grafo_lista Class Reference

Inheritance diagram for Grafo_lista:

Collaboration diagram for Grafo_lista:

Public Member Functions

- **Grafo_lista** ()

  *Construtor da classe **Grafo** (*p. **??**) lista.*
- ∼**Grafo_lista** ()

  *Destrutor da classe **Grafo** (*p. **??**) lista.*
- void **insereVertice** (float val) override
- void **insereAresta** (int origem, int destino, float val) override
- void **removeAresta** (int i, int j) override
- void **removeVertice** (int id) override
- **NodeVertex** ∗ **getVertice** (int id) override
- **NodeEdge** ∗ **getAresta** (int origem, int destino) override

Public Member Functions inherited from Grafo

- bool **eh_direcionado** ()

  Retorna se o grafo é direcionado ou não.
- bool **verticePonderado** ()

  Retorna se os vértices do grafo são ponderados ou não.
- bool **arestaPonderada** ()

  Retorna se as arestas do grafo são ponderadas ou não.
- int **getOrdem** ()

  Retorna a ordem do grafo.
- void **setOrdem** (int val)

  Define a ordem do grafo.
- void **setDirecionado** (bool val)

  Define se o grafo é direcionado ou não.
- void **setVerticePonderado** (bool val)

  Define se o grafo possui vértices ponderados.
- void **setArestaPonderada** (bool val)

  Define se o grafo possui arestas ponderadas.
- int **getGrau** ()

  Retorna o grau do grafo.
- bool **eh_completo** ()

  Retorna se o grafo é completo.
- void **carregaGrafo** (string grafo)

  Carrega o grafo na estrutura.
- void **imprimeGrafo** (string grafo)

  Imprime o grafo de acordo com a descrição do trabalho.
- float **maiorMenorDistancia** (int ponto1, int ponto2)

  Calcula a menor distância entre dois pontos utilizando o algoritmo de Dijkstra.
- int **coloracaoArestaGuloso** ()

  Algoritmo guloso que retorna a menor quantidade de cores para colorir todas as arestas do grafo de forma que não tenha duas arestas adjacentes com a mesma cor.
- int **coloracaoArestaRandomizado** ()

  Algoritmo randomizado que retorna a menor quantidade de cores para colorir todos os vértices do grafo de forma que não tenha dois vértices adjacentes com a mesma cor.
- int **coloracaoArestaReativo** ()

  Algoritmo randomizado reativo que retorna a menor quantidade de cores para colorir todas as arestas do grafo de forma que não tenha duas arestas adjacentes com a mesma cor.
- void **descoloreGrafo** ()

  Descolore todas as arestas, redefinindo seus valores para -1.

5.2.1 Member Function Documentation

5.2.1.1 getAresta()

```
NodeEdge * Grafo_lista::getAresta (
            int origem,
            int destino )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.2.1.2 getVertice()

```
NodeVertex * Grafo_lista::getVertice (
            int id )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.2.1.3 insereAresta()

```
void Grafo_lista::insereAresta (
            int origem,
            int destino,
            float val )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.2.1.4 insereVertice()

```
void Grafo_lista::insereVertice (
            float val )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.2.1.5 removeAresta()

```
void Grafo_lista::removeAresta (
            int i,
            int j )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.2.1.6 removeVertice()

```
void Grafo_lista::removeVertice (
            int id )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

The documentation for this class was generated from the following files:

- include/Grafo_lista.h
- src/Grafo_lista.cpp

5.3 Grafo_matriz Class Reference

Inheritance diagram for Grafo_matriz:

Collaboration diagram for Grafo_matriz:

Public Member Functions

- **Grafo_matriz** ()

  Construtor da classe matriz.
- ∼**Grafo_matriz** ()

  Destrutor da classe matriz.
- void **insereVertice** (float val) override
- void **insereAresta** (int origem, int destino, float val) override
- void **removeAresta** (int origem, int destino) override
- void **removeVertice** (int id) override
- **NodeVertex** ∗ **getVertice** (int id) override
- **NodeEdge** ∗ **getAresta** (int origem, int destino) override

Public Member Functions inherited from Grafo

- bool **eh_direcionado** ()

  Retorna se o grafo é direcionado ou não.
- bool **verticePonderado** ()

  Retorna se os vértices do grafo são ponderados ou não.
- bool **arestaPonderada** ()

  Retorna se as arestas do grafo são ponderadas ou não.
- int **getOrdem** ()

  Retorna a ordem do grafo.
- void **setOrdem** (int val)

  Define a ordem do grafo.
- void **setDirecionado** (bool val)

  Define se o grafo é direcionado ou não.
- void **setVerticePonderado** (bool val)

  Define se o grafo possui vértices ponderados.
- void **setArestaPonderada** (bool val)

  Define se o grafo possui arestas ponderadas.
- int **getGrau** ()

  Retorna o grau do grafo.
- bool **eh_completo** ()

  Retorna se o grafo é completo.
- void **carregaGrafo** (string grafo)

  Carrega o grafo na estrutura.
- void **imprimeGrafo** (string grafo)

  Imprime o grafo de acordo com a descrição do trabalho.
- float **maiorMenorDistancia** (int ponto1, int ponto2)

  Calcula a menor distância entre dois pontos utilizando o algoritmo de Dijkstra.
- int **coloracaoArestaGuloso** ()

  Algoritmo guloso que retorna a menor quantidade de cores para colorir todas as arestas do grafo de forma que não tenha duas arestas adjacentes com a mesma cor.

- int **coloracaoArestaRandomizado** ()

  Algoritmo randomizado que retorna a menor quantidade de cores para colorir todos os vértices do grafo de forma que não tenha dois vértices adjacentes com a mesma cor.
- int **coloracaoArestaReativo** ()

  Algoritmo randomizado reativo que retorna a menor quantidade de cores para colorir todas as arestas do grafo de forma que não tenha duas arestas adjacentes com a mesma cor.
- void **descoloreGrafo** ()

  Descolore todas as arestas, redefinindo seus valores para -1.

5.3.1 Member Function Documentation

5.3.1.1 getAresta()

```
NodeEdge * Grafo_matriz::getAresta (
            int origem,
            int destino )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.3.1.2 getVertice()

```
NodeVertex * Grafo_matriz::getVertice (
            int id )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.3.1.3 insereAresta()

```
void Grafo_matriz::insereAresta (
            int origem,
            int destino,
            float val )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.3.1.4 insereVertice()

```
void Grafo_matriz::insereVertice (
            float val )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.3.1.5 removeAresta()

```
void Grafo_matriz::removeAresta (
            int origem,
            int destino )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

5.3.1.6 removeVertice()

```
void Grafo_matriz::removeVertice (
            int id )  [override], [virtual]
```

@inheritDoc

Implements **Grafo** (p. **??**).

The documentation for this class was generated from the following files:

- include/Grafo_matriz.h
- src/Grafo_matriz.cpp

5.4 Linked_list< NodeType > Class Template Reference

Classe template para criação do grafo por lista encadeada.

```
#include <Linked_list.hpp>
```

Public Member Functions

- **Linked_list** ()
  Construtor da Lista Encadeada.
- ~**Linked_list** ()
  Destrutor da Lista encadeada.
- int **getTam** ()
  Retorna o tamanho da lista.
- NodeType * **getNodeById** (int val)
  Retorna o nó correspondente ao id passado.
- NodeType * **getUltimo** ()
  Retorna o ultimo nó
- NodeType * **getPrimeiro** ()
  Retorna o primeiro nó
- void **insereFinal** (float val)
  Insere nó no final da lista.
- void **imprimeLista** ()
  Imprime a Lista encadeada.
- void **removeNode** (NodeType *no)
  Remove o nó passado como parâmetro da lista.

Protected Member Functions

- void **limpaNodes** ()

  Função auxiliar que deleta todos os nós.

Protected Attributes

- NodeType * **primeiro**
- NodeType * **ultimo**

5.4.1 Detailed Description

template<typename NodeType>
class Linked_list< NodeType >

Classe template para criação do grafo por lista encadeada.

Template Parameters

NodeType	Tipo de nó que será utilizado

5.4.2 Member Function Documentation

5.4.2.1 getNodeById()

```
template<typename NodeType >
NodeType *  Linked_list< NodeType >::getNodeById (
             int val )
```

Retorna o nó correspondente ao id passado.

Parameters

val	Representa o id do nó a ser passado

5.4.2.2 getPrimeiro()

```
template<typename NodeType >
NodeType *  Linked_list< NodeType >::getPrimeiro ( )
```

Retorna o primeiro nó

Returns

Primeiro nó

5.4.2.3 getTam()

```
template<typename NodeType >
int  Linked_list< NodeType >::getTam ( )
```

Retorna o tamanho da lista.

Returns

Tamanho da lista

5.4.2.4 getUltimo()

```
template<typename NodeType >
NodeType *  Linked_list< NodeType >::getUltimo ( )
```

Retorna o ultimo nó

Returns

Último nó

5.4.2.5 insereFinal()

```
template<typename NodeType >
void  Linked_list< NodeType >::insereFinal (
            float val )
```

Insere nó no final da lista.

Parameters

val	Valor a ser inserido no nó

5.4.2.6 removeNode()

```
template<typename NodeType >
void  Linked_list< NodeType >::removeNode (
            NodeType * no )
```

Remove o nó passado como parâmetro da lista.

Parameters

no	Nó a ser removido

The documentation for this class was generated from the following file:

- include/Linked_list.hpp

5.5 Linked_Vertex Class Reference

Inheritance diagram for Linked_Vertex:

Collaboration diagram for Linked_Vertex:

Public Member Functions

- **Linked_Vertex** ()
  Construtor da classe Lista encadeada para os vértices.
- ∼**Linked_Vertex** ()
  Destrutor da classe lista encadeada para os vértices.
- void **insereAresta** (int origem, int destino, float val)
  Insere uma aresta entre dois vértices.
- void **removeAresta** (int i, int j)
  Remove uma aresta entre dois vértices.
- void **removeVertice** (int id)
  Remove um vértice.

Public Member Functions inherited from Linked_list< NodeVertex >

- **Linked_list** ()
  Construtor da Lista Encadeada.
- ∼**Linked_list** ()
  Destrutor da Lista encadeada.
- int **getTam** ()
  Retorna o tamanho da lista.
- **NodeVertex** ∗ **getNodeById** (int val)
  Retorna o nó correspondente ao id passado.
- **NodeVertex** ∗ **getUltimo** ()
  Retorna o ultimo nó
- **NodeVertex** ∗ **getPrimeiro** ()
  Retorna o primeiro nó
- void **insereFinal** (float val)
  Insere nó no final da lista.
- void **imprimeLista** ()
  Imprime a Lista encadeada.
- void **removeNode** (**NodeVertex** ∗no)
  Remove o nó passado como parâmetro da lista.

Additional Inherited Members

Protected Member Functions inherited from Linked_list< NodeVertex >

- void **limpaNodes** ()
  Função auxiliar que deleta todos os nós.

Protected Attributes inherited from Linked_list< NodeVertex >

- **NodeVertex * primeiro**
- **NodeVertex * ultimo**

5.5.1 Member Function Documentation

5.5.1.1 insereAresta()

```
void Linked_Vertex::insereAresta (
            int origem,
            int destino,
            float val )
```

Insere uma aresta entre dois vértices.

Parameters

origem	Define o nó em que será inserido a aresta
destino	Define o id do nó que a aresta corresponde
val	Define o peso da aresta

5.5.1.2 removeAresta()

```
void Linked_Vertex::removeAresta (
            int i,
            int j )
```

Remove uma aresta entre dois vértices.

Parameters

i	Nó em que será removida a aresta
j	Id do nó que a aresta corresponde

5.5.1.3 removeVertice()

```
void Linked_Vertex::removeVertice (
            int id )
```

Remove um vértice.

Parameters

id	Nó que será removido

The documentation for this class was generated from the following files:

- include/Linked_Vertex.h
- src/Linked_Vertex.cpp

5.6 Node Class Reference

Inheritance diagram for Node:

Public Member Functions

- **Node** ()
  Construtor da classe.
- ∼**Node** ()
  Destrutor da classe.
- **Node** ∗ **getProx** ()
  Retorna o próximo nó
- void **setProx** (**Node** ∗prox)
  Define o próximo nó
- void **setValue** (float value)
  Define o valor do nó
- float **getValue** ()
  Retorna o valor do nó
- void **setId** (int val)
  Define o id do nó
- int **getId** ()
  Retorna o id do nó

5.6.1 Member Function Documentation

5.6.1.1 getId()

```
int Node::getId ( )
```

Retorna o id do nó

Returns

ID do nó

5.6.1.2 getProx()

```
Node * Node::getProx ( )
```

Retorna o próximo nó

Returns

Próximo nó

5.6.1.3 getValue()

```
float Node::getValue ( )
```

Retorna o valor do nó

Returns

Valor do nó

5.6.1.4 setId()

```
void Node::setId (
            int val )
```

Define o id do nó

Parameters

val	Id do nó

5.6.1.5 setProx()

```
void Node::setProx (
             Node * prox )
```

Define o próximo nó

Parameters

prox	Nó que será inserido como próximo nó

5.6.1.6 setValue()

```
void Node::setValue (
            float value )
```

Define o valor do nó

Parameters

value	valor do nó

The documentation for this class was generated from the following files:

- include/Node.h
- src/Node.cpp

5.7 NodeEdge Class Reference

Inheritance diagram for NodeEdge:

Collaboration diagram for NodeEdge:

Public Member Functions

- **NodeEdge** ()
- ∼**NodeEdge** ()
- float **getPeso** ()
  Retorna o peso da aresta.
- void **setPeso** (float val)
  Define o peso da aresta.
- int **getCor** ()
  Retorna a cor do nó
- void **setCor** (int val)
  Define a cor do nó

Public Member Functions inherited from Node

- **Node** ()
  Construtor da classe.
- ∼**Node** ()
  Destrutor da classe.
- **Node** ∗ **getProx** ()
  Retorna o próximo nó
- void **setProx** (**Node** ∗prox)
  Define o próximo nó
- void **setValue** (float value)
  Define o valor do nó
- float **getValue** ()
  Retorna o valor do nó
- void **setId** (int val)
  Define o id do nó
- int **getId** ()
  Retorna o id do nó

5.7.1 Constructor & Destructor Documentation

5.7.1.1 NodeEdge()

```
NodeEdge::NodeEdge ( )
```

@inheritDoc

5.7.1.2 ~NodeEdge()

```
NodeEdge::~NodeEdge ( )
```

@inheritDoc

5.7.2 Member Function Documentation

5.7.2.1 getCor()

```
int NodeEdge::getCor ( )
```

Retorna a cor do nó

Returns

Cor

5.7.2.2 getPeso()

```
float NodeEdge::getPeso ( )
```

Retorna o peso da aresta.

Returns

Peso

5.7.2.3 setCor()

```
void NodeEdge::setCor (
            int val )
```

Define a cor do nó

Parameters

val	Cor

Returns

void

5.7.2.4 setPeso()

```
void NodeEdge::setPeso (
            float val )
```

Define o peso da aresta.

Parameters

val	Peso da aresta

The documentation for this class was generated from the following files:

- include/NodeEdge.h
- src/NodeEdge.cpp

5.8 NodeVertex Class Reference

Inheritance diagram for NodeVertex:

Collaboration diagram for NodeVertex:

Public Member Functions

- **NodeVertex** ()
- ∼**NodeVertex** ()
- **Linked_list**< **NodeEdge** > ∗ **getArestas** ()
  Retorna a lista de arestas do vértice.
- int **getGrau** ()
  Retorna o tamanho da lista de arestas do vértice (seu grau)
- void **setGrau** (int val)
  Define o grau do vértice.

Public Member Functions inherited from Node

- **Node** ()
  Construtor da classe.
- ∼**Node** ()
  Destrutor da classe.
- **Node** ∗ **getProx** ()
  Retorna o próximo nó
- void **setProx** (**Node** ∗prox)
  Define o próximo nó
- void **setValue** (float value)
  Define o valor do nó
- float **getValue** ()
  Retorna o valor do nó
- void **setId** (int val)
  Define o id do nó
- int **getId** ()
  Retorna o id do nó

5.8.1 Constructor & Destructor Documentation

5.8.1.1 NodeVertex()

```
NodeVertex::NodeVertex ( )
```

@inheritDoc

5.8.1.2 ~NodeVertex()

```
NodeVertex::~NodeVertex ( )
```

@inheritDoc

5.8.2 Member Function Documentation

5.8.2.1 getArestas()

```
Linked_list< NodeEdge > * NodeVertex::getArestas ( )
```

Retorna a lista de arestas do vértice.

Returns

Lista encadeada de arestas

5.8.2.2 getGrau()

```
int NodeVertex::getGrau ( )
```

Retorna o tamanho da lista de arestas do vértice (seu grau)

Returns

Grau do vértice

5.8.2.3 setGrau()

```
void NodeVertex::setGrau (
            int val )
```

Define o grau do vértice.

Parameters

val	Novo grau do vértice

The documentation for this class was generated from the following files:

- include/NodeVertex.h
- src/NodeVertex.cpp

Chapter 6

File Documentation

6.1 Grafo.h

```
00001 #ifndef GRAFO_H
00002 #define GRAFO_H
00003 #include "NodeVertex.h"
00004 #include "NodeEdge.h"
00005
00006
00007 class Grafo
00008 {
00009     private:
00010         int ordem = 0;
00011         bool direcionado, verticePeso, arestaPeso;
00012
00018         string imprmeSimNao(bool valor);
00019
00024         string retornaMaiorMenorDistancia();
00025
00026     public:
00027
00028     virtual ~Grafo() = default;
00034     virtual NodeVertex* getVertice(int id) = 0;
00035
00042     virtual NodeEdge* getAresta(int origem, int destino) = 0;
00043
00048     virtual void insereVertice(float val) = 0;
00049
00056     virtual void insereAresta(int origem, int destino, float val) = 0;
00057
00062     virtual void removeVertice(int id) = 0;
00063
00069     virtual void removeAresta(int origem, int destino) = 0;
00070
00075     bool eh_direcionado();
00076
00081     bool verticePonderado();
00082
00087     bool arestaPonderada();
00088
00093     int getOrdem();
00094
00099     void setOrdem(int val);
00100
00105     void setDirecionado(bool val);
00106
00111     void setVerticePonderado(bool val);
00112
00117     void setArestaPonderada(bool val);
00118
00123     int getGrau();
00124
00129     bool eh_completo();
00130
00135     void carregaGrafo(string grafo);
00136
00140     void imprimeGrafo(string grafo);
00141
00148     float maiorMenorDistancia(int ponto1, int ponto2);
00149
00154     int coloracaoArestaGuloso();
```

```
00155
00160     int coloracaoArestaRandomizado();
00161
00166     int coloracaoArestaReativo();
00167
00172     void descoloreGrafo();
00173 };
00174
00175
00176 #endif
```

6.2 Grafo_lista.h

```
00001 #ifndef GRAFO_LISTA_H
00002 #define GRAFO_LISTA_H
00003
00004 #include "Grafo.h"
00005 #include "Linked_Vertex.h"
00006 #include "NodeEdge.h"
00007 #include "NodeVertex.h"
00008
00009 class Grafo_lista : public Grafo
00010 {
00011     private:
00012         Linked_Vertex* Vertice;
00013     public:
00017         Grafo_lista();
00021         ~Grafo_lista();
00022
00026         void insereVertice(float val) override;
00027
00031         void insereAresta(int origem, int destino, float val) override;
00032
00036         void removeAresta(int i, int j) override;
00037
00041         void removeVertice(int id) override;
00042
00046         NodeVertex* getVertice(int id) override;
00047
00051         NodeEdge* getAresta(int origem, int destino) override;
00052 };
00053
00054
00055 #endif
```

6.3 Grafo_matriz.h

```
00001 #ifndef GRAFO_MATRIZ_H
00002 #define GRAFO_MATRIZ_H
00003
00004 #include "Grafo.h"
00005 #include "NodeEdge.h"
00006 #include "NodeVertex.h"
00007
00008 class Grafo_matriz : public Grafo
00009 {
00010     private:
00011         NodeEdge*** matriz_adjacencia;
00012         NodeVertex* vertices;
00013         int capacidade;
00014
00018         void inicializaMatriz();
00019
00023         void inicializaPesoVertices();
00024
00031         NodeEdge** retornaCelulaMatriz(int i, int j);
00032
00037         void resize(int novaCapacidade);
00038
00039     public:
00040
00044         Grafo_matriz();
00045
00049         ~Grafo_matriz();
00050
00054         void insereVertice(float val) override;
00055
00059         void insereAresta(int origem, int destino, float val) override;
00060
```

```
00064         void removeAresta(int origem, int destino) override;
00065
00069         void removeVertice(int id) override;
00070
00074         NodeVertex* getVertice(int id) override;
00075
00079         NodeEdge* getAresta(int origem, int destino) override;
00080 };
00081
00082
00083 #endif
```

6.4 Linked_list.hpp

```
00001 #ifndef LINKED_LIST_HPP
00002 #define LINKED_LIST_HPP
00003
00004
00010 template <typename NodeType>
00011 class Linked_list
00012 {
00013 protected:
00014     NodeType* primeiro, *ultimo;
00015
00019     void limpaNodes();
00020 private:
00021     int n;
00022 public:
00026     Linked_list();
00030     ~Linked_list();
00031
00036     int getTam();
00037
00042     NodeType* getNodeById(int val);
00043
00048     NodeType* getUltimo();
00049
00054     NodeType* getPrimeiro();
00055
00060     void insereFinal(float val);
00061
00065     void imprimeLista();
00066
00071     void removeNode(NodeType* no);
00072 };
00073
00074 #include "Linked_list.tpp"
00075
00076 #endif
```

6.5 Linked_Vertex.h

```
00001     #ifndef LINKED_VERTEX_H
00002     #define LINKED_VERTEX_H
00003
00004     #include "Linked_list.hpp"
00005     #include "NodeVertex.h"
00006
00007     class Linked_Vertex : public Linked_list<NodeVertex>
00008     {
00009         public:
00013         Linked_Vertex();
00014
00018         ~Linked_Vertex();
00019
00026         void insereAresta(int origem, int destino, float val);
00027
00033         void removeAresta(int i, int j);
00034
00039         void removeVertice(int id);
00040     };
00041
00042
00043 #endif
```

6.6 Node.h

```
00001 #ifndef NODE_H
00002 #define NODE_H
00003
00004 class Node
00005 {
00006 private:
00007     float value;
00008     Node* prox;
00009     int id;
00010
00011 public:
00015     Node();
00016
00020     ~Node();
00021
00026     Node* getProx();
00027
00032     void setProx(Node* prox);
00033
00038     void setValue(float value);
00039
00044     float getValue();
00045
00050     void setId(int val);
00051
00056     int getId();
00057 };
00058
00059
00060 #endif
```

6.7 NodeEdge.h

```
00001 #ifndef NODEEDGE_H
00002 #define NODEEDGE_H
00003 #include "Node.h"
00004
00005 class NodeEdge : public Node
00006 {
00007
00008     private:
00009     float peso;
00010     int cor;
00011
00012     public:
00016     NodeEdge();
00017
00021     ~NodeEdge();
00022
00027     float getPeso();
00028
00033     void setPeso(float val);
00034
00039     int  getCor();
00045     void setCor(int val);
00046
00047 };
00048
00049
00050 #endif
```

6.8 NodeVertex.h

```
00001 #ifndef NODEVERTEX_H
00002 #define NODEVERTEX_H
00003
00004 #include "Node.h"
00005 #include "Linked_list.hpp"
00006 #include "NodeEdge.h"
00007
00008 class NodeVertex : public Node
00009 {
00010
00011     private:
00012     Linked_list<NodeEdge>* Arestas;
00013     int grau;
00014
```

```
00015     public:
00019     NodeVertex();
00020
00024     ~NodeVertex();
00025
00030     Linked_list<NodeEdge>* getArestas();
00031
00036     int getGrau();
00037
00042     void setGrau(int val);
00043 };
00044
00045
00046 #endif
```