Documentation

This implementation uses the C++ language.

We shall define:

- class DirectedGraph – representation of the directed graph
- class UI – testing purposes over the DirectedGraph class

DirectedGraph class documentation:

The DirectedGraph class serves as the core structure for representing and manipulating directed graphs within the system. It encapsulates a comprehensive suite of methods for graph management, including vertex and edge operations, as well as graph analysis utilities.

1) Initialization and Configuration:

- Constructor:

`DirectedGraph()` – initializes a graph instance
`DirectedGraph(int nrVertices)` – initializes a graph instance with a specified number of vertices and edges.
`DirectedGraph(const DirectedGraph& g)` – copy constructor

- Destructor:
`~DirectedGraph()` – destroys the instance

2) Getters and setters:

```
unsigned long get_nr_of_vertices() const – Returns the number of vertices of the graph

unsigned long get_nr_of_edges() const – Returns the number of edges of the graph

std::vector<std::pair<int, int>> get_edges() const – Returns all the vertices of the graph
```

```
std::vector<std::pair<int, int>> get_edges() const - Returns all the edges of the graph

std::vector<int> get_inbound_vertices(int v) - Returns all the inbound vertices of a specific vertex

std::vector<int> get_outbound_vertices(int v) - Returns all the outbound vertices of a specific vertex

unsigned long get_in_degree(int v) - Returns the in degree of a vertex

unsigned long get_out_degree(int v) - Returns the out degree of a vertex

long long get_cost(int v1, int v2) - Returns the cost of an edge

void set_cost(int v1, int v2, int c) - Sets the cost of an edge
```

3) Vertex and Edge manipulation:

```
bool add_vertex(int v) - Adds a vertex to the graph

bool remove_vertex(int v) - Removes a vertex from the graph

bool add_edge(int v1, int v2, int c) - Adds an edge to the graph

bool remove_edge(int v1, int v2) - Removes an edge from the graph
```

4) Graph analysis utilities:

```
bool is_vertex(int v) const - Checks if a vertex already exists

bool is_edge(int v1, int v2) - Checks if an edge already exists
```

5) Other fuctions:

```
void clear() - Removes all vertices and edges

void generate_random_graph(int nrv, int nre) - Generates a random graph
```

This documentation outlines the functionalities provided by the **Graph** class for constructing and manipulating directed graphs. Through its methods, users can effectively manage graph elements, perform analyses, and explore graph structures in various computational contexts.

UI class documentation:

The UI class is the entry point for users to interact with the Graph Management System, providing a comprehensive suite of functionalities for managing directed graphs. This document outlines the functionalities available through the UI class.

1) Initialization:

- Constructor:

```
UI() – initializes a UI instance
```

- Destructor:

```
~UI() – destroys a UI instance
```

2) Print functions:

```
void printMenu() – prints the menu

void printNrOfVertices() – prints the number of vertices

void printNrOfEdges() – prints the number of edges

void printVertices() – prints the vertices

void printEdges() – prints the edges

void printDegreesOfVertices() – prints the in degree and the
out degree of a vertex

void printInboundEdgesOfVertex() – prints the inbound edges
of a vertex

void printOutboundEdgesOfVertex() – prints the outbound edges
of a vertex

void retrieveCostOfEdge() – prints a specific edge and it’s
cost

void printCopyGraph() – makes a deep copy a DirectedGraph and
prints it after it eliminates the first vertex
```

3) Vertex and Edge manipulation:

```
void addVertex() - Adds a vertex to the graph

void removeVertex() - Removes a vertex from the graph

void addEdge() - Adds an edge to the graph

void removeEdge() - Removes an edge from the graph
```

4) Other graph utilities functions:

```
void createRandomGraph() - creates a random graph

void seeIfIsEdge() - tells us if there is a specific edge

void modifyCostOfEdge() - modifies the cost of an edge
```

5) File operation functions:

```
void readGraphFromFile() - reads a graph from a file

void writeGraphInFile() - writes a graph in a file
```

This documentation provides a roadmap for navigating the UI functionalities of the Graph Management System, designed to offer intuitive access to complex graph operations and analyses.