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 DirectedGraphclass 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

1. 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

1. 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

1. 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

1. 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

1. 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

1. 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

1. 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

1. 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.