Graph Class Reference

`#include <Graph.h>`

Public Member Functions

int	**get_nr_of_vertices** () const
int	**get_nr_of_edges** () const
std::map< node, std::vector< node > >	**get_dict_in** ()
std::map< node, std::vector< node > >	**get_dict_out** ()
std::map< node, node >	**get_vertices** ()
std::map< edge, int >	**get_dict_cost** ()
auto	**get_dict_in_iterator** (node vertex)
auto	**get_dict_out_iterator** (node vertex)
std::map< edge, int >::iterator	**get_dict_cost_iterator** ()
auto	**get_vertices_iterator** ()
void	**read_from_file** (const std::string &)
void	**write_to_file** (const std::string &file_name)
bool	**is_edge** (node n1, node n2)
int	**in_degree** (node vertex)
int	**out_degree** (node)
void	**modify_cost** (node n1, node n2, int cost)
int	**get_cost** (node n1, node n2)
void	**delete_edge** (node n1, node n2)
void	**add_edge** (node n1, node n2, int cost)
void	**add_node** (node n)
void	**delete_node** (node n)
	Graph (const **Graph** &g)
bool	**is_vertex** (node n)

Static Public Member Functions

static **Graph**	**random_graph** (int n, int m)

Detailed Description

Directed **Graph** class with 3 maps.

Constructor & Destructor Documentation

◆ Graph()

Graph::Graph (const **Graph** & g) inline

Copy constructor for the **Graph** class

Parameters

g the graph to be copied

Member Function Documentation

◆ add_edge()

```
void Graph::add_edge ( node n1,
                       node n2,
                       int  cost
                     )
```

Function that adds an edge to the graph

Parameters

n1 the first node from the edge

n2 the second node from the edge

cost the cost of the new edge

Exceptions

if edge already exists or if nodes are invalid

O(1)

◆ add_node()

```
void Graph::add_node ( node n )
```

Function that adds a new node to the graph

Parameters

n the new node to be added

Exceptions

if node already exists

O(1)

◆ delete_edge()

```
void Graph::delete_edge ( node  n1,
                          node  n2
                        )
```

Function that deletes an existing edge

Parameters

n1 the first node from the edge

n2 the second node from the edge

Exceptions

if edge (n1, n2) doesn't exist

O(n)

◆ delete_node()

```
void Graph::delete_node ( node  n )
```

Function that deletes an existing node from the graph, along with all edges connecting it to other edges

Parameters

n the node to be deleted

Exceptions

if node doesn't exist

O(n)

◆ get_cost()

```
int Graph::get_cost ( node  n1,
                      node  n2
                    )
```

Function that retrieves the cost of an edge

Parameters

n1 the first node from the edge

n2 the second node from the edge

Exceptions

if edge (n1, n2) doesn't exist

Returns

the cost of the edge (n1, n2) (int)

Theta(1)

in_degree()

int Graph::in_degree (node vertex)

Function that returns the in-degree of a given vertex

Parameters

vertex an existing node (int)

Exceptions

if vertex doesn't exist

Returns

the in-degree of the vertex (int)

Theta(1)

is_edge()

bool Graph::is_edge (node n1,
node n2
)

Function that checks if two nodes are part of an edge.

Parameters

n1 first vertex (int)

n2 second vertex (int)

Returns

true if edge exists else false

O(logn)

is_vertex()

bool Graph::is_vertex (node n)

Function that verifies if a given node index is in the graph

Parameters

n the node to search for

Returns

true if the node exists, false otherwise

O(n)

modify_cost()

```
void Graph::modify_cost ( node  n1,
                          node  n2,
                          int   cost
                        )
```

Function that modifies the cost of a given edge

Parameters

n1 first vertex

n2 second vertex

cost the new cost of the edge

Exceptions

if edge (n1, n2) doesn't exist in the graph

Theta(1)

◆ out_degree()

```
int Graph::out_degree ( node   )
```

Function that returns the out-degree of a given vertex

Parameters

vertex an existing node (int)

Exceptions

if vertex doesn't exist

Returns

the out-degree of the vertex (int)

Theta(1)

◆ random_graph()

static **Graph** Graph::random_graph (int n,
int m
) static

Static function that creates a random graph

Parameters

n the number of vertices of the new graph

m the number of edges of the new graph

Exceptions

if graph cannot be composed (the number of edges is too big for how many vertices are)

Returns

the new graph

◆ read_from_file()

void Graph::read_from_file (const std::string &)

Function that reads a graph from a file.

Exceptions

if file doesn't exist

Theta(n)

◆ write_to_file()

void Graph::write_to_file (const std::string & file_name)

Function that writes a graph to a given file.

Parameters

file_name file path (std::string)

Exceptions

if file cannot be opened for writing

Theta(n)

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

- **Graph.h**