# Graphs laboratory 2 homework Documentation

Rapeanu George - Alexandru

# **CONTENTS:**

	pyth		1		
	1.1	UI module	1		
	1.2	UndirectedGraph module	1		
	1.3	UndirectedGraphTests module	4		
2 Indices and tables					
Рy	thon ]	Module Index	9		
In	dex		11		

#### **CHAPTER**

## **ONE**

## **PYTHON**

## 1.1 UI module

```
UI.display_edges (edges)
This function displays a given list of edges

Parameters edges (list) - list of edges represented as tuples

Returns None

UI.display_graph (graph)
displays a graph

Parameters graph (UndirectedGraph.UndirectedGraph) - the graph
Returns None

UI.display_vertices (vertices)
This function displays the given vertices

Parameters vertices (list) - the vertices

Returns None

UI.main()
The main of the program

Returns None
```

# 1.2 UndirectedGraph module

```
class UndirectedGraph.UndirectedGraph (vertices, edges)
    Bases: object

add_edge (x, y, z)
    This function adds the edge from x to y to the graph

Parameters

• x (str) - the first vertex

• y (str) - the second vertex

• z (int) - the cost
```

#### Raises

 $\bullet$  Exception-if types do not follow the specification

```
• Exception – if nodes do not exist
```

• Exception – if edge already exists

#### add\_vertex(x)

This function adds the vertex x to the graph

**Parameters**  $\mathbf{x}$  (str) – the vertex to be added

#### Raises

- Exception if x is not string
- Exception if x already exists

#### copy()

This function retrieves a copy of the current graph

Returns a Graph copy

#### get\_degree (x)

This function returns the degree of a vertex

**Parameters**  $\mathbf{x}$  (str) – the vertex

**Returns** the in degree of the vertex x

Raises Exception – if x doesn't exist

#### $get_edge_cost(x, y)$

This function returns the cost of the edge between x and y

#### **Parameters**

- $\mathbf{x}(str)$  the first vertex
- $\mathbf{y}(str)$  the second vertex

**Returns** the cost of the edge from x to y

Raises Exception – if there is no edge from x to y

#### has\_vertex(vertex)

This function returns true if the provided vertex exists, false otherwise

**Parameters** vertex (str) – the vertex

Returns boolean

### $is\_edge(x, y)$

This function returns True if the edge x-y exists, false otherwise

#### **Parameters**

- $\mathbf{x}$  (str) the first vertex
- $\mathbf{y}(str)$  the second vertex

Returns True if an edge exists, false otherwise

**Raises** Exception – if x or y are not vertices

#### $modify\_edge\_cost(x, y, z)$

This function modifies the cost of the edge from x to y

#### **Parameters**

•  $\mathbf{x}$  (str) – the first vertex

2 Chapter 1. python

- **y** (str) the second vertex
- **z** (int) the new cost

Raises Exception – if there is no edge from x to y

#### parse\_adjacent\_edges (x)

This function returns an iterable of deepcopied vertices

**Parameters**  $\mathbf{x}$  – the vertex for which to retrieve the iterator

Returns iterator to a deepcopied list of outbound vertices

Raises Exception - if the vertex doesn't exist

#### parse\_vertices()

This function returns an iterable containing nodes

The nodes are deepcopied, in order to avoid being modified from the outside :return: iterator through a list of deepcopied nodes

#### $remove\_edge(x, y)$

This function removes the edge from x to y from the graph

#### **Parameters**

- $\mathbf{x}$  (str) the first vertex
- $\mathbf{y}(str)$  the second vertex

**Raises** Exception – if edge already exists

#### $remove\_vertex(x)$

This function removes the vertex x from the graph

**Parameters**  $\mathbf{x}$  (str) – the vertex to be removed

**Raises** Exception – if x doesn't exist

#### UndirectedGraph.get\_connected\_components(graph)

Returns a list of UndirectedGraph-s representing the connected component of the given graph

Parameters graph (UndirectedGraph) - the graph

Returns list of UndirectedGraph

#### UndirectedGraph.random\_graph (n, m)

This function creates a random graph with specified number of vertices and edges

#### **Parameters**

- n (int) the number of vertices
- m (int) the number of edges

Returns a graph with specified parameters

**Raises** Exception – if invalid parameters

UndirectedGraph.read\_graph(filename)

#### **This function reads a graph from a file.** It supports 2 formats .txt and .modified.txt

In case of .txt, the file is supposed to look like this:

On the first line, the number n of vertices and the number m of edges; On each of the following m lines, three numbers, x, y and c, describing an edge.

In case of .modified.txt, the file is supposed to look like this:

On the first line, the number n of vertices and the number m of edges On the second line, a list of the n vertices separated by space On each of the following m lines, three numbers, x, y and c, describing an edge.

**Parameters filename** (str) – the file from which to read(name, relative path or absolute path)

Returns Graph

Raises Exception - in case of invalid format

UndirectedGraph.write\_graph (filename, graph)

This function writes a graph from a file. It supports 1 format .modified.txt

On the first line, the number n of vertices and the number m of edges On the second line, a list of the n vertices separated by space On each of the following m lines, three numbers, x, y and c, describing an edge.

#### **Parameters**

- filename (str) the filename to which to read(name, relative path or absolute path),
   MUST end in .modified.txt
- graph (UndirectedGraph) the graph to be written

Raises Exception - if invalid data

# 1.3 UndirectedGraphTests module

```
class UndirectedGraphTests.UndirectedGraphTests (methodName='runTest')
    Bases: unittest.case.TestCase
    setUp()
        Hook method for setting up the test fixture before exercising it.
    test_add_edge()
    test_add_vertex()
    test_constructor()
    test_copy()
    test eq()
    test_get_connected_components()
    test_get_degree()
    test_get_edge_cost()
    test_has_vertex()
    test_is_edge()
    test_modify_edge_cost()
    test_parse_adjacent_edges()
    test_parse_vertices()
    test random graph()
    test_read_graph()
```

4 Chapter 1. python

```
test_remove_edge()
test_remove_vertex()
test_write_graph()
```

6 Chapter 1. python

## **CHAPTER**

# TWO

# **INDICES AND TABLES**

- genindex
- modindex
- search

Graphs	laboratory	2 homework	Documentation

# **PYTHON MODULE INDEX**

## u

UI, 1 UndirectedGraph, 1 UndirectedGraphTests, 4

10 Python Module Index

# **INDEX**

A	R
<pre>add_edge() (UndirectedGraph.UndirectedGraph</pre>	<pre>random_graph() (in module UndirectedGraph), 3 read_graph() (in module UndirectedGraph), 3 remove_edge() (UndirectedGraph.UndirectedGraph</pre>
C	remove_vertex() (Undirected- Graph.UndirectedGraph method), 3
copy () (UndirectedGraph.UndirectedGraph method), 2	S
D display_edges() (in module UI), 1 display_graph() (in module UI), 1 display_vertices() (in module UI), 1	setUp() (UndirectedGraphTests method), 4  T
G  get_connected_components() (in module UndirectedGraph), 3  get_degree() (UndirectedGraph.UndirectedGraph	test_add_edge() (UndirectedGraphT- ests.UndirectedGraphTests method), 4  test_add_vertex() (UndirectedGraphT- ests.UndirectedGraphTests method), 4  test_constructor() (UndirectedGraphT-
<pre>method), 2 get_edge_cost() (Undirected- Graph.UndirectedGraph method), 2</pre>	ests.UndirectedGraphTests method), 4  test_copy() (UndirectedGraphTests.UndirectedGraphTests method), 4  test_eq() (UndirectedGraphTests.UndirectedGraphTest
H has_vertex() (UndirectedGraph.UndirectedGraph method), 2	ests. Undirected Graph Tests method), 4 test_get_connected_components() (Undirect- ed Graph Tests. Undirected Graph Tests method),  4
is_edge()	test_get_degree() (UndirectedGraphT- ests.UndirectedGraphTests method), 4  test_get_edge_cost() (UndirectedGraphT- ests.UndirectedGraphTests method), 4
M main() (in module UI), 1 modify_edge_cost() (Undirected- Graph.UndirectedGraph method), 2	test_has_vertex() (UndirectedGraphT- ests.UndirectedGraphTests method), 4  test_is_edge() (UndirectedGraphT- ests.UndirectedGraphTests method), 4  test_modify_edge_cost() (UndirectedGraphT- ests.UndirectedGraphTests method), 4
P parse_adjacent_edges() (Undirected- Graph.UndirectedGraph method), 3 parse_vertices() (Undirected- Graph.UndirectedGraph method), 3	test_parse_adjacent_edges() (Undirected- GraphTests.UndirectedGraphTests method), 4 test_parse_vertices() (UndirectedGraphTests.UndirectedGraphTests method), 4

#### **Graphs laboratory 2 homework Documentation**

```
test_random_graph()
                               (UndirectedGraphT-
        ests. Undirected Graph Tests method), 4
test_read_graph()
                               (UndirectedGraphT-
        ests. Undirected Graph Tests\ method),\ 4
                               (UndirectedGraphT-
test_remove_edge()
        ests. Undirected Graph Tests method), 4
                               (UndirectedGraphT-
test_remove_vertex()
        ests.UndirectedGraphTests method), 5
                               (UndirectedGraphT-
test_write_graph()
        ests. Undirected Graph Tests method), 5
U
UI (module), 1
UndirectedGraph (class in UndirectedGraph), 1
UndirectedGraph (module), 1
UndirectedGraphTests (class in Undirected-
        GraphTests), 4
UndirectedGraphTests (module), 4
W
write_graph() (in module UndirectedGraph), 4
```

12 Index