Graphs lab3 Documentation

Rapeanu George - Alexandru

CONTENTS:

1	python				
	1.2	GraphTests module	4		
	1.3	UI module	5		
2 Indices and tables					
Python Module Index					
Inc	dex		11		

CHAPTER

ONE

PYTHON

1.1 Graph module

```
class Graph (vertices, edges)
      Bases: object
      add_edge(x, y, z)
           This function adds the edge from x to y to the graph
               Parameters
                    • \mathbf{x}(str) – the first vertex
                    • \mathbf{y}(str) – the second vertex
                    • z (int) – the cost
               Raises
                    • 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_edge_cost(x, y)
           This function returns the cost of the edge from x to y
               Parameters
                    • \mathbf{x}(str) – the first vertex
```

• **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

get_in_degree(x)

This function returns the in 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_out_degree (x)

This function returns the out degree of a vertex

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

Returns the out degree of the vertex x

Raises Exception – if x doesn't exist

$is_edge(x, y)$

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

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

parse_inbound_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 inbound vertices

Raises Exception – if the vertex doesn't exist

parse_outbound_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

2 Chapter 1. python

$remove_edge(x, y)$

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

Graph.floyd_warshall(graph, u, v)

This function returns a tuple containing on the first position the smallest cost, and on the second position the walk from u to v which achieves this cost. Returns (inf,[]) if there is no walk

Parameters

- **graph** (Graph) the graph
- **u** (str) the first node
- $\mathbf{v}(str)$ the second node

Returns tuple

Graph.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

Graph.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: the origin, the target and the cost of that 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: the origin, the target and the cost of that 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

Graph.write_graph (filename, graph)

1.1. Graph module

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: the origin, the target and the cost of that edge.

Parameters

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

Raises Exception – if invalid data

1.2 GraphTests module

```
class GraphTests.GraphTests (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_floyd_warshall()
    test_get_edge_cost()
    test_get_in_degree()
    test_get_out_degree()
    test_is_edge()
    test_modify_edge_cost()
    test_parse_inbound_edges()
    test_parse_outbound_edges()
    test_parse_vertices()
    test_random_graph()
    test_read_graph()
    test_remove_edge()
    test_remove_vertex()
    test_write_graph()
```

4 Chapter 1. python

1.3 UI module

${\tt UI.display_edges}\ (edges)$

This function displays a given list of edges

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

Returns None

UI.display_vertices (vertices)

This function displays the given vertices

Parameters vertices (list) – the vertices

Returns None

UI.main()

The main of the program

Returns None

1.3. Ul module 5

6 Chapter 1. python

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

g

Graph, 1
GraphTests, 4

u

UI,5

10 Python Module Index

INDEX

A	<pre>remove_vertex() (Graph.Graph method), 3</pre>		
<pre>add_edge() (Graph.Graph method), 1 add_vertex() (Graph.Graph method), 1</pre>	S		
С	setUp() (GraphTests.GraphTests method), 4		
copy() (Graph.Graph method), 1	•		
D	test_add_edge() (<i>GraphTests.GraphTests method</i>), 4		
display_edges() (in module UI), 5	test_add_vertex() (GraphTests.GraphTests		
display_vertices() (in module UI),5	method), 4		
F	test_constructor() (GraphTests.GraphTests method), 4		
floyd_warshall() (in module Graph), 3	test_copy() (GraphTests.GraphTests method), 4 test_eq() (GraphTests.GraphTests method), 4		
G	test_floyd_warshall() (GraphTests.GraphTests method), 4		
<pre>get_edge_cost() (Graph.Graph method), 1 get_in_degree() (Graph.Graph method), 2</pre>	test_get_edge_cost() (GraphTests.GraphTests method), 4		
<pre>get_out_degree() (Graph.Graph method), 2 Graph (class in Graph), 1</pre>	<pre>test_get_in_degree() (GraphTests.GraphTests method), 4</pre>		
Graph (module), 1 GraphTests (class in GraphTests), 4 GraphTests (module), 4	<pre>test_get_out_degree() (GraphTests.GraphTests method), 4</pre>		
Graphiteses (mounte), 4	test_is_edge() (GraphTests.GraphTests method), 4		
	test_modify_edge_cost() (GraphT-ests.GraphTests method), 4		
is_edge() (Graph.Graph method), 2	test_parse_inbound_edges() (GraphT-ests.GraphTests method), 4		
M	test_parse_outbound_edges() (GraphT-		
main() (in module UI), 5	ests.GraphTests method), 4		
<pre>modify_edge_cost() (Graph.Graph method), 2</pre>	<pre>test_parse_vertices() (GraphTests.GraphTests method), 4</pre>		
P parse_inbound_edges() (Graph.Graph method),	test_random_graph() (GraphTests.GraphTests method), 4		
2	test_read_graph() (GraphTests.GraphTests		
parse_outbound_edges() (Graph.Graph method),	<pre>method), 4 test_remove_edge() (GraphTests.GraphTests</pre>		
parse_vertices() (<i>Graph.Graph method</i>), 2	method), 4		
R	test_remove_vertex() (GraphTests.GraphTests method), 4		
<pre>random_graph() (in module Graph), 3 read_graph() (in module Graph), 3 remove_edge() (Graph.Graph method), 2</pre>	test_write_graph() (GraphTests.GraphTests method), 4		

Graphs lab3 Documentation

U

UI (module), 5

W

write_graph() (in module Graph), 3

12 Index