Graphs lab 4 Documentation

Rapeanu George - Alexandru

May 18, 2022

CONTENTS:

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

- **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 **`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 **`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

- **`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

`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

- **`x`** (*`str`*) – the first vertex
- **`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

- **`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_adjacent_edges(*x*)

This function returns an iterable of deepcopied vertices

Parameters 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

- **x** (*str*) – the first vertex
- **y** (*str*) – the second vertex

Raises Exception – if edge already exists

remove_vertex(*x*)

This function removes the vertex x from the graph

Parameters 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.**get_minimum_spanning_tree**(*graph*)

Given a graph, this function returns the MST of it using Kruskal's algorithm

Parameters graph (*UndirectedGraph*) – the graph

Returns the tree as an 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_MST`()

`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()

test_remove_edge()

test_remove_vertex()

test_write_graph()

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX