Documentation for Practical work 1

Specification

We shall define a class named Graph representing a directed graph.

The class Graph will provide the following methods:

def dict_cost(self):

returns the dictionary in which are stored the edges and costs of the graph

def dict_in(self):

returns the dictionary in which are stored the inbound vertices of the graph

def dict_out(self):

returns the dictionary in which are stored the outbound vertices of the graph

def number_of_vertices(self):

returns the number of vertices of the graph

def number_of_edges(self):

returns the number of edges of the graph

def parse_vertices(self):

iterator for the vertices of the graph

def parse_inbound(self, x):

iterator for the inbound vertices of the graph

def parse_outbound(self, x):

iterator for the outbound vertices of the graph

def parse_cost(self):

iterator for the edges of the graph and their cost

def add_vertex(self, x):

adds a new vertex to the graph and returns true if added , false otherwise

the new vertex x shouldn’t already exist

def remove_vertex(self, x):

    removes the vertex ‘x’ from the graph and returns true if removed, false otherwise

    the vertex x should exist

def in_degree(self, x):

    returns the in degree of the given vertex ‘x’

    the vertex x should exist

def out_degree(self, x):

    returns the out degree of the given vertex ‘x’

    the vertex x should exist

def add_edge(self, x, y, cost):

    adds a new edge to the graph and returns true if added and false otherwise

    the edge shouldn’t already exist

def remove_edge(self, x, y):

    removes a edge of the graph and returns true if removed and false otherwise

    the edge should already exist

def is_edge(self, x, y):

    returns the cost of the edge (x,y) if it exists and false otherwise

def is_vertex(self,x):

    returns true if the vertex exists in the graph and false otherwise

def change_cost(self, x, y, cost):

    changes the cost of the edge (x,y) to the new cost and returns true if changed , false otherwise

    the edge (x,y) should already exist

def make_copy(self):

    returns a copy of the graph

Implementation

The implementation of the class Graph is the following:

self._number_of_vertices – the number of vertices in the graph

self._number_of_edges - the number of edges in the graph

self._dict_in = {} – the dictionary that stores the inbound vertices for each vertex of the graph

(each vertex is the key and the inbound vertices of that vertex stored as a list is the value)

self._dict_out = {} – the dictionary that stores the outbound vertices for each vertex of the graph

(each vertex is the key and the outbound vertices of that vertex stored as a list is the value)

self._dict_cost = {} – the dictionary that stores all the edges of the graph and their cost

(each edge stored as a tuple is the key and the cost is the value)

There are 2 additional functions for reading and writing a graph from/to a file:

def write_graph_to_text_file(graph,file_name)

Take as parameters the graph that is going to be written and the name of the destination file.

If the file dosen't exist a new one is created.

def read_graph_from_text_file(file_name)

Takes as parameter the name of the source file. The file must already exist