**GRAPHS 1**

**def add\_edge(self, source, destination, cost):**

Adds an edge to the graph

Checks if the edge is in the graph, and if not, adds it

Params:

source: the source vertix

destination: destination vertix

cost: the cost of the edge

**def remove\_edge(self, source, destination):**

Removes an edge

Checks whether the edge is in the graph, and if so, it removes it

Params:

source: the source vertex

destination: destination vertex

**def remove\_vertix(self, vertix):**

"""

Removes a vertix

Checks if a vertix is in the graph, and if so, it removes it and all the edges associated to it

Params:

vertix: the vertix to be removed

**def read\_graph(self, file\_name):**

"""

Reads a graph from a file

Reads a graph from a file and adds all the edges and vertices from it

Params:

file\_name: the name of the file

"""

**def read\_modified(self, file\_name):**

"""

Reads a graph from a file

Reads a graph from a file with a different format and adds all the edges and vertices

Params:

file\_name: the name of the file

"""

**def save\_graph(self, file\_name):**

"""

Saves a graph to a file

Writes the graph to a file, with the isolated vertices on one line, and all the edges on separate lines

Params:

file\_name: the name of the file

"""

**def parse\_vertices(self):**

"""

Returns the list of vertices

Returns the list of vertices to be parsed

"""

**def check\_for\_edge(self, source, destination):**

"""

Checks whether and edge exists or not

Checks whether there is an edge between two vertices

Params:

source: the source vertix

destination: the destination vertix

Returns: true if there is an edge, false otherwise

"""

**def get\_out\_edges(self, vertex):**

"""

Get the outbound edges from a vertix

Gets the outbound edges from a vertix after it checks if it is in the graph

Params:

vertex: the vertex we check

Returns: an iterator containing the outbound edges

"""

**def get\_in\_edges(self, vertex):**

"""

Get the inbound edges from a vertix

Gets the inbound edges from a vertix after it checks if it is in the graph

Params:

vertex: the vertex we check

Returns: an iterator containing the inbound edges

"""

**def add\_vertix(self, vertex, mode=""):**

"""

Adds a vertix to the graph

Adds a vertix to the graph if it is not already inside

Params:

vertex: the vertex to be added

mode: the mode for adding(ui or from reading a graph)

"""

**def get\_edge\_cost(self, edge):**

"""

Returns the cost of an edge

Returns the cost of an edge if it is in the graph

Params:

edge: the edge we check

Returns: the cost

"""

**def modify\_edge\_cost(self, edge, new\_value):**

"""

Modifies the cost of an edge

Modifies the cost of an edge if it is in the graph

Params:

edge: the edge we modify

new\_value: the new cost

"""

**def get\_degree(self, vertex):**

"""

Returns the degrees of an vertex

Returns the in and out degrees of a given vertex

Params:

vertex: the vertex we check

Returns: a tuple containing the outbound and inbound degrees

"""

**def generate\_random\_graph(vertex\_count, edge\_count):**

"""

Generates a random graph

Generatex a random graph with a given nr. of vertices and edges

Params:

vertex\_count: the number of vertices

edge\_count: the number of edges

"""