Practical work, number 1 – documentation

Specifications

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

The class Graph will provide the following methods:

```
def add_edge(self, source, target, cost):
```

Adds and edge from the source to the target with the given cost.

Condition: the edge doesn’t exist.

```
def remove_edge(self, source, target):
```

Removes the edge from the source to the target.

```
def add_vertex(self, vertex):
```

Adds a new vertex to the graph.

```
def remove_vertex(self, vertex):
```

Removes the given vertex.

```
def get_set_of_vertices(self):
```

Returns a list of the set of vertices.

```
def is_edge(self, source, target):
```

Returns True if there is an edge from source to target and False in contrary.

```
def in_degree_of_vertex(self, vertex):
```

Returns the in degree of a vertex (the length of the in neighbours list).

```
def out_degree_of_vertex(self, vertex):
```

Returns the out degree of a vertex (the length of the out neighbours list).

```
def in_neighbours(self, vertex):
```

Returns a list of the in neighbours of the given vertex.

```
def out_neighbours(self, vertex):
```

Returns a list of the out neighbours of the given vertex.

```
def cost_of_an_edge(self, source, target):
```

Returns the cost of an edge.

```
def set_new_cost(self, source, target, new_cost):
```

Sets a new cost on the edge between the source and the target.

```
def number_of_vertices(self):
```

Returns the number of vertices.

```
def number_of_edges(self):
```

Returns the number of edges.

We shall define another graph called Controller.

The class Controller will have the following methods:

```
def create_graph_from_file(self):
```

Reads a graph from a file and it returns it.

```
def save_graph_to_file(self)
```

Saves a graph to a file.

```
def generate_random_graph(self, vertices, edges)
```

Generates and returns a random graph.

```
def save_graph(self, v, e):
```

Saves a randomly generated graph to a file.

Implementation

The implementation uses the classes Graph, Controller and UI.

The class Graph will have the following data members:

- din: a dictionary for the inbound edges of each vertex
- dout: a dictionary for the outbound edges of each vertex
- dcost: a dictionary for each edge's cost
- vertices: the number of vertices
- edges: the number of edges

The class Controller will have the following data members:

- graph: the graph which the user is working on, read from the file

The class UI will have the following data members:

- controller: the controller of the application