

Laboratory 1

Documentation

This project is implemented in the Python language. To solve the problem statement we need to implement a Graph class which represents a direct graph and an UI class that represents the menu that operates the methods of the Graph.

Implementation:

Graph class has a set of vertices and three dictionaries representing the inbound and outbound neighbours of a vertex and one for the cost of an edge.

The set of vertices, `self.__vertices`, keeps track of every vertex of the graph.

The dictionary, `self.__outbound_neighbours`, keeps a set of every neighbour of a given vertex `x`. For example `self.__outbound_neighbours[x]` is a set containing all the vertices that form an edge from `x` to that vertex.

For the dictionary `self.__inbound_neighbours` is the same explanation as above but it keeps track the inbound neighbours of a given vertex `x`.

Graph class has the following methods:

- Constructor -> creates the graph
- `vertices_iterator` -> returns an iterator to the set of vertices
- `outbound_neighbours_iterator` -> returns an iterator to the set of outbound neighbours of a vertex
- `inbound_neighbours_iterator` -> returns an iterator to the set of inbound neighbours of a vertex
- `edges_iterator` -> returns an iterator to the set of edges
- `is_vertex` -> checks if a given vertex is in the graph
- `is_edge` -> checks if a given edge is in the graph
- `count_vertices` -> returns the number of vertices in the graph

- `count_edges` -> returns the number of edges in the graph
- `in_degree` -> returns the in degree of a given vertex
- `out_degree` -> returns the out degree of a given vertex
- `get_edge_cost` -> getter for the cost of a given edge
- `set_edge_cost` -> setter for the cost of a given edge
- `add_vertex` -> adds a given vertex in the graph
- `remove_vertex` -> removes a vertex from the graph
- `add_edge` -> adds a given edge in the graph
- `remove_edge` -> removes an edge from the graph
- `copy` -> returns a deep copy of the graph

Also, there are two friend functions that help us to read a graph from a file and save a graph to a file.

- `read_file` -> creates and returns a graph from a text file
- `write_file` -> writes to the file the data of the graph