Graphs 2023 – 2024

Documentation – Practical work 1

student: Rareș-Andrei Cotoi
Computer Science, group 912

The solution I have provided for the practical work no. 1 is implemented in Python, using Layered Architecture in 4 different files:

- **domain.py** -> which stores the declaration of our data type (the dictionary associated with the graph and the costs' matrix);
- **repository.py** -> which stores the logical operations of the assignment.
- **user_interface.py** -> which stores the functions for the console-based UI and calls the appropriate functions in repository.
- **main.py** -> the main file, which calls the UI.

Since the project is a small one, there was no need to also implement a service module. The task requires that the graph is read from a text file, hence I also implemented the graph_data.txt file.

Domain.py

In the domain module, I implemented an ADT for the graph which consists of:

- **list_of_edges** -> a dictionary that stores at position [node1] all the nodes node2 with the property that there exists an edge between node1 and node2.
- **data_map** -> a matrix which stores at position [node1][node2] the associated number with the edge node1 – node2.

Also, in this module I implemented some key-operations that modify the graph's structure, such as: **add_node, add_edge, remove_node, remove_edge** etc.

Repository.py

The repository module implements the appropriate functionalities that the user selects to perform on the graph in the UI module. Here are implemented most of the "back" functionalities, such as:

- Check if there is an edge between two nodes – returns the presence of node2 in node1's array in the list_of_edges dictionary;
- Get in degree of a node – returns the number of appearances of the specified node in the other node's arrays in the list_of_edges dictionary;
- Get out degree of a node – returns the length of the specified node's array in list_of_edges;
- Get set of outbound edges of a node – returns the array at position node1 in list_of_edges;
- Get set of inbound edges of a node – creates a new array with the nodes where the specified node is present in their array;
- Modify data of a certain edge – modifies the data matrix at position [node1][node2];
- Print the graph – returns the whole current instance of the domain;
- Copy the graph – creates a deep copy of the ADT (the deep copy makes sure that it is actually created a copy of the data, not just a reference to the initial graph that will modify the data in-block).

User_interface.py

The UI module prints the console menu and handles the user input. It also calls the appropriate functions in repo, based on the user input.

main.py

The main module calls the UI module, in order to run the program appropriately.