Documentation

This implementation uses the Python language.

We shall define:

- **class Graph** – representation of the directed graph

- **class UI** – testing purposes over the Graph class

- **functions for input/output files** (reading/writing in a text file)

Graph Class Documentation

The **Graph** class serves as the core structure for representing and manipulating directed graphs within the system. It encapsulates a comprehensive suite of methods for graph management, including vertex and edge operations, as well as graph analysis utilities.

Initialization and Configuration

- **Constructor (__init__)**: Initializes a graph instance with a specified number of vertices and edges. It prepares internal data structures to store graph information such as incoming and outgoing edges for each vertex, and costs associated with each edge.

Property Setters

- Methods to update graph properties:
    - **set_number_of_vertices(number_of_vertices)**: Updates the total number of vertices in the graph.
    - **set_number_of_edges(number_of_edges)**: Updates the total number of edges in the graph.
    - **set_incoming_edges(incoming_edges)**: Assigns a dictionary representing the incoming edges for each vertex.
    - **set_outgoing_edges(outgoing_edges)**: Assigns a dictionary representing the outgoing edges for each vertex.
    - **set_costs(costs)**: Assigns a dictionary representing the costs associated with each edge.

Data Access and Iteration

- **Iterators for Graph Elements**:
    - **parse_vertices()**: Yields each vertex in the graph.
    - **parse_inbound_edges(vertex)**: Yields all vertices that have an inbound edge to the specified vertex.
    - **parse_outbound_edges(vertex)**: Yields all vertices that the specified vertex has an outbound edge to.
    - **parse_cost()**: Yields each edge and its associated cost in the graph.

Property Getters

- Accessors for retrieving graph properties:
    - **number_of_vertices**: Returns the total number of vertices in the graph.
    - **number_of_edges**: Returns the total number of edges in the graph.
    - **incoming_edges**: Provides access to the dictionary of incoming edges.
    - **outgoing_edges**: Provides access to the dictionary of outgoing edges.
    - **costs**: Returns the dictionary mapping each edge to its cost.

Vertex and Edge Manipulation

- **Methods for Managing Graph Elements**:
    - **add_vertex(vertex)**: Attempts to add a new vertex to the graph. Ensures that the vertex does not already exist.
    - **remove_vertex(vertex)**: Attempts to remove a specified vertex and all related edges. Ensures that the vertex exists before removal.
    - **add_edge(x, y, cost)**: Adds a new edge from vertex **x** to **y** with the specified cost. Verifies that the edge does not pre-exist.
    - **remove_edge(x, y)**: Removes an existing edge from vertex **x** to **y**. Ensures the edge exists before proceeding with removal.

Graph Analysis Utilities

- **Methods for Analyzing Graph Properties**:
    - **in_degree(vertex)**: Calculates and returns the in-degree of the specified vertex.
    - **out_degree(vertex)**: Calculates and returns the out-degree of the specified vertex.
    - **find_if_edge(x, y)**: Checks if an edge from vertex **x** to **y** exists and returns its cost if found.
    - **change_cost(x, y, new_cost)**: Updates the cost of an existing edge from vertex **x** to **y** to a new value.

Graph Duplication

- **copy_graph()**: Creates and returns a deep copy of the graph instance, ensuring independence of the duplicate from the original.

This documentation outlines the functionalities provided by the **Graph** class for constructing and manipulating directed graphs. Through its methods, users can effectively manage graph elements, perform analyses, and explore graph structures in various computational contexts

File Operations:

- **write_graph_to_file(graph, filename)**: Writes the graph's structure and data to a specified file. Includes vertices, edges, and edge costs.
- **read_graph_from_file(filename)**: Constructs a graph by reading its structure and data from a specified file.

User Interface (UI) Documentation for Graph Management System

The UI class is the entry point for users to interact with the Graph Management System, providing a comprehensive suite of functionalities for managing directed graphs. This document outlines the functionalities available through the UI class.

Initialization

- **UI Class Constructor**: Initializes the UI with an empty list for storing graphs (**self._graphs**) and sets the current working graph to **None** (**self._current**).

Graph Creation and Management

- **add_empty_graph**: Adds a new, empty graph to the system and sets it as the current graph.
- **create_random_graph_ui**: Prompts the user for the number of vertices and edges and creates a graph with random edges and costs. It checks for the possibility of the requested graph before creation.
- **generate_random**: A helper method that generates a random graph given the number of vertices and edges.

Graph Switching

- **switch_graph_ui**: Allows switching between existing graphs. It lists available graphs and prompts the user to select one.

Graph Information Display

- **get_number_of_vertices_ui**: Displays the number of vertices in the current graph.

- **get_number_of_edges_ui**: Shows the number of edges in the current graph.
- **list_all_outbound**: Lists all vertices and their outbound edges.
- **list_all_inbound**: Displays all vertices and their inbound edges.
- **list_outbound**: Lists outbound edges for a specified vertex.
- **list_inbound**: Lists inbound edges for a specified vertex.
- **list_all_costs**: Displays all edges in the graph along with their costs.
- **parse_all_vertices**: Lists all vertices in the current graph.

Graph Modification

- **add_vertex_ui**: Adds a new vertex to the current graph.
- **delete_vertex_ui**: Removes a specified vertex from the current graph.
- **add_edge_ui**: Adds a new edge between specified vertices with a given cost.
- **remove_edge_ui**: Removes an edge between specified vertices.
- **modify_cost_ui**: Changes the cost of a specified edge.

Graph Analysis

- **get_in_degree_ui**: Displays the in-degree of a specified vertex.
- **get_out_degree_ui**: Shows the out-degree of a specified vertex.
- **check_if_edge_ui**: Checks if there is an edge between two specified vertices and displays its cost if it exists.

Graph Persistence

- **write_graph_to_file_ui**: Writes the current graph to a file.
- **read_graph_from_file_ui**: Loads a graph from a file and sets it as the current graph.
- **write_modified_graph_to_file_ui**: Writes modifications of the current graph to a file.
- **read_modified_graph_from_file_ui**: Reads a modified graph from a file.

Utility and Miscellaneous

- **copy_current_graph_ui**: Creates a deep copy of the current graph and adds it to the system.
- **print_menu**: Displays the main menu with all available commands to the user.
- **start**: The main loop of the UI, allowing the user to execute commands until they choose to exit.

Starting the System

- **UI().start()**: Initializes the UI and starts the main interaction loop, welcoming the user and providing them with options to manage and analyze graphs.

This documentation provides a roadmap for navigating the UI functionalities of the Graph Management System, designed to offer intuitive access to complex graph operations and analyses.