**DirectedGraph Project Documentation**

**1. Project Overview**

This project implements a directed graph structure with various methods for manipulating and interacting with the graph. The graph is represented using vertices (integers) and edges with associated costs. The project allows for graph creation, reading from a file, modifying the graph, and performing queries on the graph.

**2. Class: DirectedGraph**

**Description:**

A directed graph implementation that allows adding/removing vertices and edges, as well as querying and modifying graph data.

**Methods:**

**\_\_init\_\_(self):**

* **Purpose:** Initializes the graph with empty data structures for adjacency list, inbound list, and edge costs.
* **Input:** None
* **Output:** None

**read\_from\_file(self, filename):**

* **Purpose:** Reads the graph data from a file and populates the graph's internal structures.
* **Input:** filename (string) – The name of the file containing the graph data in the format: n m followed by source vertex, destination vertex, edge cost.
* **Output:** None

**write\_to\_file(self, filename):**

* **Purpose:** Writes the graph to a file in the same format as read\_from\_file.
* **Input:** filename (string) – The name of the file to write the graph data.
* **Output:** None

**add\_vertex(self, vertex):**

* **Purpose:** Adds a new vertex to the graph if it does not already exist.
* **Input:** vertex (integer) – The vertex to add.
* **Output:** None

**remove\_vertex(self, vertex):**

* **Purpose:** Removes the specified vertex and its associated edges from the graph.
* **Input:** vertex (integer) – The vertex to remove.
* **Output:** None
* **Error:** Raises ValueError if the vertex does not exist.

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

* **Purpose:** Adds a directed edge with a specified cost between two vertices.
* **Input:** source (integer), destination (integer), cost (integer) – The source and destination vertices, and the cost of the edge.
* **Output:** None
* **Error:** Raises ValueError if the edge already exists.

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

* **Purpose:** Removes the directed edge between the source and destination vertices.
* **Input:** source (integer), destination (integer) – The vertices of the edge to remove.
* **Output:** None
* **Error:** Raises ValueError if no edge exists between the vertices.

**get\_vertices(self):**

* **Purpose:** Returns a list of all vertices in the graph.
* **Input:** None
* **Output:** list of integers representing the vertices.

**get\_number\_of\_vertices(self):**

* **Purpose:** Returns the total number of vertices in the graph.
* **Input:** None
* **Output:** integer – The number of vertices in the graph.

**edge\_exists(self, source, destination):**

* **Purpose:** Checks if an edge exists from the source vertex to the destination vertex.
* **Input:** source (integer), destination (integer) – The vertices to check.
* **Output:** boolean – True if the edge exists, False otherwise.

**get\_edges(self):**

* **Purpose:** Prints all edges in the graph along with their costs.
* **Description:** This method iterates through the edge\_costs dictionary, printing each edge and its associated cost in the format: source -> destination : cost.
* **Input:** None
* **Output:** None (prints edges to the console)

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

* **Purpose:** Returns the in-degree (number of incoming edges) of a specified vertex.
* **Input:** vertex (integer) – The vertex to check.
* **Output:** integer – The in-degree of the vertex.
* **Error:** Raises ValueError if the vertex does not exist.

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

* **Purpose:** Returns the out-degree (number of outgoing edges) of a specified vertex.
* **Input:** vertex (integer) – The vertex to check.
* **Output:** integer – The out-degree of the vertex.
* **Error:** Raises ValueError if the vertex does not exist.

**get\_outbound\_edges(self, vertex):**

* **Purpose:** Returns a list of outbound edges (destination vertices) for a specified vertex.
* **Input:** vertex (integer) – The vertex to check.
* **Output:** list of integers – List of outbound vertices.
* **Error:** Raises ValueError if the vertex does not exist.

**get\_inbound\_edges(self, vertex):**

* **Purpose:** Returns a list of inbound edges (source vertices) for a specified vertex.
* **Input:** vertex (integer) – The vertex to check.
* **Output:** list of integers – List of inbound vertices.
* **Error:** Raises ValueError if the vertex does not exist.

**get\_edge\_cost(self, source, destination):**

* **Purpose:** Returns the cost of an edge between two vertices.
* **Input:** source (integer), destination (integer) – The source and destination vertices.
* **Output:** integer – The cost of the edge, or None if the edge does not exist.

**set\_edge\_cost(self, source, destination, cost):**

* **Purpose:** Sets the cost of a specified edge.
* **Input:** source (integer), destination (integer), cost (integer) – The source and destination vertices and the new cost.
* **Output:** None
* **Error:** Raises ValueError if the edge does not exist.

**create\_new\_graph(self):**

* **Purpose:** Creates a deep copy of the current graph.
* **Input:** None
* **Output:** A new DirectedGraph instance (deep copy of the current graph)

**generate\_random\_graph(nr\_vertices, nr\_edges):**

* **Purpose:** Generates a random directed graph with a specified number of vertices and edges.
* **Input:** nr\_vertices (integer) – The number of vertices, nr\_edges (integer) – The number of edges.
* **Output:** A new DirectedGraph instance with randomly generated edges.

**get\_outbound\_edges\_iterator(self, vertex):**

* **Purpose:** Returns an iterator over the outbound edges of a specified vertex.
* **Input:** vertex (integer) – The vertex to iterate.
* **Output:** Iterator over outbound edges (source, destination).

**get\_inbound\_edges\_iterator(self, vertex):**

* **Purpose:** Returns an iterator over the inbound edges of a specified vertex.
* **Input:** vertex (integer) – The vertex to iterate.
* **Output:** Iterator over inbound edges (source, destination).

**3. Class: UI**

**Description:**

Handles the user interface for interacting with the directed graph.

**Methods:**

**\_\_init\_\_(self):**

* **Purpose:** Initializes the UI and loads the graph from a file.
* **Input:** None
* **Output:** None

**print\_menu(self):**

* **Purpose:** Displays the menu of options for the user to choose from.
* **Input:** None
* **Output:** None

**menu(self):**

* **Purpose:** Main loop for handling user input and executing graph operations based on the selected option.
* **Input:** None
* **Output:** None