DirectedGraph Class Specification

Constructor & Destructor:

- `def __init__(self, n: int = 0, m: int = 0) -> None:` Constructs a directed graph with `n` vertices and `m` edges.

Member Functions:

- `def add_vertex(self, v: int) -> None:` Adds a new vertex `v` to the graph.
  - Raises `Exception` if `v` is invalid or already exists.
- `def remove_vertex(self, v: int) -> None:` Removes vertex `v` and all associated edges.
  - Raises `Exception` if `v` is invalid or does not exist.
- `def add_edge(self, v_from: int, v_to: int, e_cost: int) -> None:` Adds a directed edge from `v_from` to `v_to` with the given cost.
  - Raises `Exception` if the edge is invalid or already exists.
- `def remove_edge(self, v_from: int, v_to: int) -> None:` Removes the directed edge from `v_from` to `v_to`.
  - Raises `Exception` if the edge does not exist.
- `def get_nr_of_vertices(self) -> int:` Returns the number of vertices in the graph.
- `def get_nr_of_edges(self) -> int:` Returns the number of edges in the graph.
- `def vertices_iterator(self) -> Iterator[int]:` Returns an iterator for all vertices in the graph.
- `def edges_iterator(self) -> Iterator[list]:` Returns an iterator for all edges along with their costs.
- `def inbound_iterator(self, vertex_to: int) -> Iterator[int]:` Returns an iterator for all inbound edges of vertex `vertex_to`.
- `def outbound_iterator(self, vertex_from: int) -> Iterator[int]:` Returns an iterator for all outbound edges of vertex `vertex_from`.
- `def get_in_degree(self, vertex_to: int) -> int:` Returns the in-degree of vertex `vertex_to`.
- `def get_out_degree(self, vertex_from: int) -> int:` Returns the out-degree of vertex `vertex_from`.
- `def get_cost(self, vertex_from: int, vertex_to: int) -> int:` Returns the cost of the edge from `vertex_from` to `vertex_to`.
  - Raises `Exception` if the edge does not exist.

- `def set_cost(self, vertex_from: int, vertex_to: int, new_cost: int) -> None`: Changes the cost of an existing edge.
    - Raises `Exception` if the edge does not exist.
- `def copy(self) -> 'DirectedGraph'`: Returns a deep copy of the graph.
- `def existing_vertex(self, v: int) -> bool`: Checks if v exists in the graph.
- `def valid_vertex(self, v: int) -> bool`: Checks if v is a valid vertex.
- `def valid_edge(self, v_from: int, v_to: int) -> bool`: Checks if an edge from `v_from` to `v_to` is valid.
- `def existing_edge(self, v_from: int, v_to: int) -> bool`: Checks if an edge from `v_from` to `v_to` exists in the graph.
- `def get_endpoints(self, edge_id: int) -> tuple[int, int]`: Returns the endpoints of the given edge ID.
- `def clear(self) -> None`: Clears all vertices and edges from the graph.

I/O Functions:

- `def generate_random_graph(nr_of_vertices:int, nr_of_edges:int): -> 'DirectedGraph':Generates a random directed graph with a specified number of vertices and edges.`
- `def read_graph_from_file(self, file_descriptor:str) -> None:Reads the directed graph from a file.`
- `def save_graph_to_file(self, file_descriptor:str): -> None:Saves the directed graph to a file.`

Edge Class Specification

Constructor:

- `def __init__(self, v1: int, v2: int) -> None`: Constructs an edge between v1 and v2 and computes its unique ID.

Member Functions:

- `def get_edge_id(self) -> int`: Returns the unique edge ID.