Aarav Dev

Description
Problem 9.3 asks to write a program to perform a topological sort on a graph.

I wrote the code in a singular Main class, which contains the topologicalSort method. Here is a description of the algorithm:

1. Initialize a queue to store vertices with no incoming edges and a list to store the sorted vertices.
2. Compute the in-degree of each vertex in the graph.
3. Enqueue all vertices with in-degree of 0 into the queue.
4. While the queue is not empty, dequeue a vertex from the queue, add it to the sorted list, and decrement the in-degree of its neighbors.
5. If any neighbor of the dequeued vertex now has an in-degree of 0, enqueue it into the queue.
6. Repeat steps 4 and 5 until the queue is empty.
7. If the number of vertices in the sorted list is equal to the number of vertices in the graph, return the sorted list. Otherwise, the graph contains a cycle and the method returns an empty list.

My method takes in two parameters: an integer representing the number of courses and a 2D int array as the prerequisite graph. This represents a graph in university structure where a directed edge (v, w) indicates that course v must be completed before course w may be attempted. A topological ordering of these courses is any course sequence that does not violate the prerequisite requirement.

The main method creates a test 2D int array representing the prerequisites and sets the number of courses and then runs the topological sort on these variables.

Complexity Analysis
The time complexity of the topological sort algorithm is O(V + E), where V is the number of vertices and E is the number of edges in the graph.

This is because each vertex and each edge is visited exactly once, and the operations performed on each vertex and edge are constant time. Specifically, we compute the in-degree of each vertex in O(E) time, and then we perform a traversal of the graph in O(V + E) time.

Therefore, the time complexity of the algorithm is O(V + E).