Course Schedule II LeetCode

There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [a_i, b_i] indicates that you **must** take course b_i first if you want to take course a_i.

- For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.

Return *the ordering of courses you should take to finish all courses*. If there are many valid answers, return **any** of them. If it is impossible to finish all courses, return **an empty array**.

Example:

prerequisites = {{1, 0}, {2, 0}, {3, 1}, {3, 2}}

The given Prerequisite courses:

To take course 1 you have to complete course 0 first.

To take course 2 you have to complete course 0 first.

To take course 3 you have to complete course 1 first.

To take course 3 you have to complete course 2 first.

The Order to complete courses:

0 1 2 3

Example 2:

prerequisites = {{1, 0}, {0, 1}}

To take course 1 you have to complete course 0 first.

To take course 0 you have to complete course 1 first.

It's not possible to complete all courses

Approach 1: Function to find the order of courses to complete based on prerequisites

- **Explanation:**
    - The **findOrder** function constructs an adjacency list representing prerequisites and calculates in-degrees for each course.
    - It then performs a BFS traversal, updating in-degrees and enqueuing courses with in-degree 0.
    - The order in which courses are completed is stored in the **topologicalSort** vector.

    - If the topological sort size is equal to the number of courses, it is possible to complete all courses.

- **Time Complexity:**
    - **The time complexity is O(V + E), where V is the number of courses, and E is the number of prerequisites.**
        - **Both the creation of the adjacency list and BFS traversal contribute to the time complexity.**
- **Space Complexity:**
    - **The space complexity is O(V + E), where V is the number of courses, and E is the number of prerequisites.**
        - **The adjacency list, in-degrees vector, BFS queue, and the topological sort vector contribute to space usage.**