2023-04-30 - Handout – Graphs

# Q1. Find if Path Exists in Graph

Link: <https://leetcode.com/problems/find-if-path-exists-in-graph/>

There is a **bi-directional** graph with n vertices, where each vertex is labeled from 0 to n - 1 (**inclusive**). The edges in the graph are represented as a 2D integer array edges, where each edges[i] = [ui, vi] denotes a bi-directional edge between vertex ui and vertex vi. Every vertex pair is connected by **at most one** edge, and no vertex has an edge to itself.

You want to determine if there is a **valid path** that exists from vertex source to vertex destination.

Given edges and the integers n, source, and destination, return true*if there is a****valid path****from*source*to*destination*, or*false*otherwise.*

|  |  |
| --- | --- |
| **Example 1:**    **Input: n = 3, edges = [[0,1],[1,2],[2,0]], source = 0, destination = 2**  **Output: true**  **Explanation: There are two paths from vertex 0 to vertex 2:**  **- 0 → 1 → 2**  **- 0 → 2** | **Example 2:**    **Input: n = 6, edges = [[0,1],[0,2],[3,5],[5,4],[4,3]], source = 0, destination = 5**  **Output: false**  **Explanation: There is no path from vertex 0 to vertex 5.** |

# Constraints:

# 1 <= n <= 2 \* 105

# 0 <= edges.length <= 2 \* 105

# edges[i].length == 2

# 0 <= ui, vi <= n - 1

# ui != vi

# 0 <= source, destination <= n - 1

# There are no duplicate edges.

# There are no self edges.

# Q2. Course Schedule II

Link: <https://leetcode.com/problems/course-schedule-ii/>

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] = [ai, bi] indicates that you **must** take course bi first if you want to take course ai.

* 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 1:**  **Input: numCourses = 2, prerequisites = [[1,0]]**  **Output: [0,1]**  **Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0. So the correct course order is [0,1].** | **Example 2:**  **Input: numCourses = 4, prerequisites = [[1,0],[2,0],[3,1],[3,2]]**  **Output: [0,2,1,3]**  **Explanation: There are a total of 4 courses to take. To take course 3 you should have finished both courses 1 and 2. Both courses 1 and 2 should be taken after you finished course 0.**  **So one correct course order is [0,1,2,3]. Another correct ordering is [0,2,1,3].** | **Example 3:**  **Input: numCourses = 1, prerequisites = []**  **Output: [0]** |

# Constraints:

# 1 <= numCourses <= 2000

# 0 <= prerequisites.length <= numCourses \* (numCourses - 1)

# prerequisites[i].length == 2

# 0 <= ai, bi < numCourses

# ai != bi

# All the pairs [ai, bi] are distinct.

# Q3. Count Ways to Build Rooms in an Ant Colony

Link: <https://leetcode.com/problems/count-ways-to-build-rooms-in-an-ant-colony/>

You are an ant tasked with adding n new rooms numbered 0 to n-1 to your colony. You are given the expansion plan as a **0-indexed** integer array of length n, prevRoom, where prevRoom[i] indicates that you must build room prevRoom[i] before building room i, and these two rooms must be connected **directly**. Room 0 is already built, so prevRoom[0] = -1. The expansion plan is given such that once all the rooms are built, every room will be reachable from room 0.

You can only build **one room** at a time, and you can travel freely between rooms you have **already built** only if they are **connected**. You can choose to build **any room** as long as its **previous room** is already built.

Return *the****number of different orders****you can build all the rooms in*. Since the answer may be large, return it **modulo** 109 + 7.

|  |  |
| --- | --- |
| **Example 1:**    **Input: prevRoom = [-1,0,1]**  **Output: 1**  **Explanation: There is only one way to build the additional rooms: 0 → 1 → 2** | **Example 2:**    **Input: prevRoom = [-1,0,0,1,2]**  **Output: 6**  **Explanation:**  **The 6 ways are:**  **0 → 1 → 3 → 2 → 4**  **0 → 2 → 4 → 1 → 3**  **0 → 1 → 2 → 3 → 4**  **0 → 1 → 2 → 4 → 3**  **0 → 2 → 1 → 3 → 4**  **0 → 2 → 1 → 4 → 3** |

# Constraints:

# n == prevRoom.length

# 2 <= n <= 105

# prevRoom[0] == -1

# 0 <= prevRoom[i] < n for all 1 <= i < n

# Every room is reachable from room 0 once all the rooms are built.