COMP S265F Design and Analysis of Algorithms
Lab 8: Depth-First Search and Topological Sort

In this lab, we will apply Depth First Search (DFS) and Topological Sort to solve a LeetCode problem "210. Course Schedule II": https://leetcode.com/problems/course-schedule-ii/

1. Course schedule problem

There are a total of n courses you have to take labelled from 0 to $n-1$. Some courses may have prerequisites, for example, if `prerequisites[i] = [`a_i`, `b_i`]`, this means you must take the course b_i before the course a_i.

Given the total number of courses `numCourses` and a list of the `prerequisite` pairs, 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*.

```python
class Solution:
    def findOrder(self, numCourses: int, prerequisites: List[List[int]]) -> List[int]:
```

The problem has given the following examples and constraints:

- **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 <= `a_i`, `b_i` < numCourses`
- a_i `!=` b_i
- All the pairs $[a_i, b_i]$ are distinct.

2. Problem formulation

The prerequisites can form a directed graph. If the directed graph does not contain any cycle, then the graph is a DAG (directed acyclic graph) and we can use DFS to find a valid topological sort; otherwise, it does not contain any valid solution and an empty array should be returned.

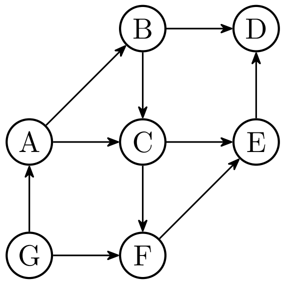
As shown in Unit 4 Slides 73 and 74, if the graph contains a cycle, there is a back edge in the Depth-First tree, where a back edge is an edge connecting a vertex to is ancestor in the DF tree. In other words, during DFS, if we discover a gray vertex through an edge, then this edge connects a vertex to its ancestor (as it was discovered but not finished yet) and is a back edge.

Your task. To solve the problem, modify the topological sort algorithm such that

- The `visited` list is changed from two states (True/False) to three states, i.e., 0 (not discovered)/ 1 (discovered but not finished)/ 2 (finished).
- The recursive function for DFS returns a boolean value, where `True` indicates that a back edge has been found, and `False` indicates that no back edge was found.

3. Exercises

Given the following directed acyclic graph (DAG):



Question 1. Perform a Breadth-First Search (BFS) on the given graph, using vertex G as the source.

(a) List the vertices in the visited order of BFS, and show for each vertex v, its distance $dist[v]$ from G .

(b) Draw the breadth-first tree obtained.

Question 2. Perform a Depth-First Search (DFS) on the given graph, using vertex G as the source.

(a) List the vertices in the discovered order of DFS and show for each vertex v, its discovery time $d[v]$ and finish time $f[v]$.

(b) Draw the depth-first tree obtained.

(c) Show the classification of each edge (tree edge, back edge, forward edge, cross edge).

Question 3. Perform a topological sort on the given graph, where we perform DFS on undiscovered vertices in alphabetical order.

(a) For each DFS in the topological sort, show its source vertex and list the vertices in the finished order.

(b) Show the ordering of vertices obtained by the topological sort.

Question 4.

(a) Design an algorithm to check whether a directed graph contains cycle.

(b) Use your algorithm in (a) to check whether the following directed graph G contains a cycle. Show your steps clearly.