

Q

207. Course Schedule

- **PatrickSUDO O** 2021-02-02
- ◆ LeetCode-Medium DFS Graph Topological Sort

LeetCode 207. Course Schedule □

Problem:

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 true if you can finish all courses. Otherwise, return false .

Example 1:

```
Input: numCourses = 2, prerequisites = [[1,0]]
Output: true
Explanation: There are a total of 2 courses to take.
To take course 1 you should have finished course 0. So it is possible.
```

Example 2:

```
Input: numCourses = 2, prerequisites = [[1,0],[0,1]]
Output: false
Explanation: There are a total of 2 courses to take.
To take course 1 you should have finished course 0, and to take course 0 you should also h
```

Patrick Su

Constraints:

• 1 <= numCourses <= 105

• 0 <= prerequisites.length <= 5000

prerequisites[i].length == 2

• 0 <= ai, bi < numCourses

• All the pairs prerequisites[i] are unique.

Solution:

Topological sorting of directed graphs. The purpose of topological sorting is to check whether it can be completed or not. If there are no loops, all the prerequisites and non-prerequisites can be

completed. First, convert the course into a 2D graph, and then build a visted array to record the

visited and the visiting. There are three states in total.

1. Haven't visited yet (0)

2. Visiting (1)

3. Visited (2)

If the node we step on is already visiting, we are back to ourselves when exploring its neighbors.

Therefore, it means we have a loop. If you go to the one we have already visited, it means we can

finish the route, and there is no loop. If there is any loop detected during the DFS , return true.

It's similar to post-order traversal, first set as visiting, then explore neighbors, enter dfs

recursively, and mark as visited after returning

Time complexity: O(V + E) than one times

Space complexity: O(V + E)

Python

Q

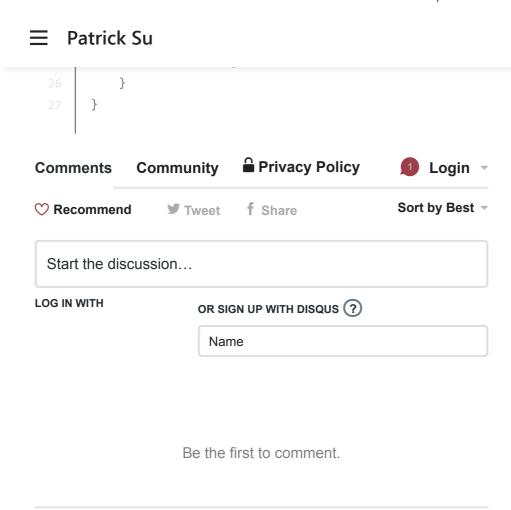
□ Patrick Su
 □

```
graph = [[] for _ in range(numCourses)]
for first, second in prerequisites:
   graph[first].append(second)
# 0 = unknown # 1 = visiting # 2 = visited
visit = [0] * numCourses
def dfs(curr, visit):
    if visit[curr] == 1: return True
    if visit[curr] == 2: return False
   visit[curr] = 1
    for j in graph[curr]:
        if dfs(j, visit): return True
   visit[curr] = 2
    return False
for i in range(numCourses) :
    if dfs(i, visit): return False
return True
```

Java

```
class Solution {
    public boolean canFinish(int numCourses, int[][] prerequisites) {
        ArrayList<Integer>[] graph = new ArrayList[numCourses];
        for(int i = 0; i < numCourses; i ++) graph[i] = new ArrayList<>();
        for(int[] course: prerequisites){
            graph[course[0]].add(course[1]);
        }
        int[] visited = new int[numCourses];
        for(int i = 0; i < numCourses; i ++){</pre>
            if(dfs(graph, visited, i)) return false;
        }
        return true;
   }
   private boolean dfs(ArrayList<Integer>[] graph, int[] visited, int curr){
        if(visited[curr] == 1) return true;
        if(visited[curr] == 2) return false;
        visited[curr] = 1;
        for(Integer neighbor : graph[curr]){
            if(dfs(graph, visited, neighbor)) return true;
```

A Do Not Call My Data



Last Updated: February 14, 2021 20:54

Q