210. Course Schedule II

Hint

Medium 10.2K 323

Companies

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 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]

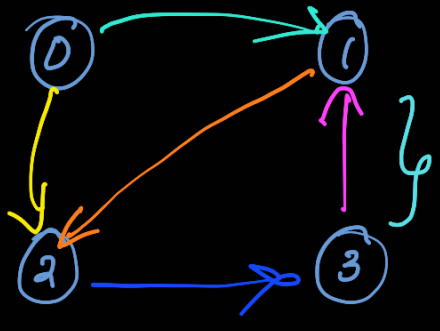
Constraints:

- `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**.

Accepted **911.4K** | Submissions **1.8M** | Acceptance Rate **49.5%**

[(1,0) , (2,0) , (2,1) , (3,2) , (1,3)]



i.e. To do course (1), you need to complete course (3)

note: we cannot interpret the edges the otherway bcoz we won't get the correct topological sort.

```
class Solution {
public:
    bool cycleDfs(vector<vector<int>> &v,int i,vector<int> &visited,vector<int> &currVisited,stack<int> &st){
        cout<<i<<endl;
        visited[i] = 1;
        currVisited[i] = 1;
        for(auto j:v[i]){
            if(!visited[j]){
                if(cycleDfs(v,j,visited,currVisited,st))
                    return true;
            }
            else if(currVisited[j] == 1)
                return true;
        }
        st.push(i);
        currVisited[i] = 0;
        return false;
    }

    vector<int> findOrder(int numCourses, vector<vector<int>>& prerequisites) {
        vector<vector<int>> v(numCourses,vector<int>(0));
        vector<int> visited(numCourses,0);
        vector<int> currVisited(numCourses,0);

        for(auto &temp:prerequisites){
```

```
        v[temp[1]].push_back(temp[0]);
    }
    stack<int> st;
    for(int i=0;i<numCourses;i++){
        if(!visited[i] && cycleDfs(v,i,visited,currVisited,st))
            return {};
    }

    vector<int> ans;

    while(!st.empty()){
        ans.push_back(st.top());
        st.pop();
    }

    return ans;
  }
};
```

v[temp[0]].push_back(temp[1]) won't work

getting Toposort

$T(n): O(m) + O(n) + O(m) + O(n)$

$S(n): O(2n) + O(n) + O(n) + O(n)$

where m: prerequisites length

n: numCourses