Question: https://leetcode.com/problems/course-schedule-ii/

Solution :https://leetcode.com/problems/course-schedule-ii/solution/

Code:

class Solution {

static int WHITE = 1;

static int GRAY = 2;

static int BLACK = 3;

boolean isPossible;

Map<Integer, Integer> color;

Map<Integer, List<Integer>> adjList;

List<Integer> topologicalOrder;

private void init(int numCourses) {

this.isPossible = true;

this.color = new HashMap<Integer, Integer>();

this.adjList = new HashMap<Integer, List<Integer>>();

this.topologicalOrder = new ArrayList<Integer>();

// By default all vertces are WHITE

for (int i = 0; i < numCourses; i++) {

this.color.put(i, WHITE);

}

}

private void dfs(int node) {

// Don't recurse further if we found a cycle already

if (!this.isPossible) {

return;

}

// Start the recursion

this.color.put(node, GRAY);

// Traverse on neighboring vertices

for (Integer neighbor : this.adjList.getOrDefault(node, new ArrayList<Integer>())) {

if (this.color.get(neighbor) == WHITE) {

this.dfs(neighbor);

} else if (this.color.get(neighbor) == GRAY) {

// An edge to a GRAY vertex represents a cycle

this.isPossible = false;

}

}

// Recursion ends. We mark it as black

this.color.put(node, BLACK);

this.topologicalOrder.add(node);

}

public int[] findOrder(int numCourses, int[][] prerequisites) {

this.init(numCourses);

// Create the adjacency list representation of the graph

for (int i = 0; i < prerequisites.length; i++) {

int dest = prerequisites[i][0];

int src = prerequisites[i][1];

List<Integer> lst = adjList.getOrDefault(src, new ArrayList<Integer>());

lst.add(dest);

adjList.put(src, lst);

}

// If the node is unprocessed, then call dfs on it.

for (int i = 0; i < numCourses; i++) {

if (this.color.get(i) == WHITE) {

this.dfs(i);

}

}

int[] order;

if (this.isPossible) {

order = new int[numCourses];

for (int i = 0; i < numCourses; i++) {

order[i] = this.topologicalOrder.get(numCourses - i - 1);

}

} else {

order = new int[0];

}

return order;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>