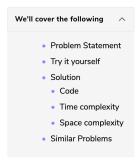


Grokking the Coding Interview: Patterns for **Coding Questions** 93% completed Q Search Course Introduction Pattern: Sliding Window Pattern: Two Pointers Pattern: Fast & Slow pointers Pattern: Merge Intervals Pattern: Cyclic Sort Pattern: In-place Reversal of a LinkedList Pattern: Tree Breadth First Search Pattern: Tree Depth First Search Pattern: Two Heaps Pattern: Subsets Pattern: Modified Binary Search Pattern: Bitwise XOR Pattern: Top 'K' Elements Pattern: K-way merge Pattern: 0/1 Knapsack (Dynamic Programming) Pattern: Topological Sort (Graph) Topological Sort (medium) Tasks Scheduling (medium) Tasks Scheduling Order (medium) All Tasks Scheduling Orders

Alien Dictionary (hard)

Tasks Scheduling (medium)



Problem Statement

There are 'N' tasks, labeled from '0' to 'N-1'. Each task can have some prerequisite tasks which need to be completed before it can be scheduled. Given the number of tasks and a list of prerequisite pairs, find out if it is possible to schedule all the tasks.

Example 1:

```
Input: Tasks=3, Prerequisites=[0, 1], [1, 2]
Output: true
Explanation: To execute task '1', task '0' needs to finish first. Similarly, task '1' needs t
o finish
before '2' can be scheduled. A possible sceduling of tasks is: [0, 1, 2]
```

Example 2:

```
Input: Tasks=3, Prerequisites=[0, 1], [1, 2], [2, 0]
Output: false
Explanation: The tasks have cyclic dependency, therefore they cannot be sceduled.
```

Example 3:

```
Input: Tasks=6, Prerequisites=[2, 5], [0, 5], [0, 4], [1, 4], [3, 2], [1, 3]
Output: true
Explanation: A possible sceduling of tasks is: [0 1 4 3 2 5]
```

Try it yourself

Try solving this question here:

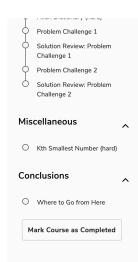
Solution

This problem is asking us to find out if it is possible to find a topological ordering of the given tasks. The tasks are equivalent to the vertices and the prerequisites are the edges.

We can use a similar algorithm as described in Topological Sort to find the topological ordering of the tasks. If the ordering does not include all the tasks, we will conclude that some tasks have cyclic dependencies.

Code

Here is what our algorithm will look like (only the highlighted lines have changed):







```
G C++
     class TaskScheduling {
       public static boolean isSchedulingPossible(int tasks, int[][] prerequisites) {
          List<Integer> sortedOrder = new ArrayList<>();
          if (tasks <= 0)
          \label{lambdap} HashMap < Integer> in Degree = new HashMap < (); // count of incoming edges for every vertex \\ HashMap < Integer>. List < Integer> graph = new HashMap < (); // adjacency list graph \\ \\
             inDegree.put(i, 0);
             graph.put(i, new ArrayList<Integer>());
          for (int i = 0; i < prerequisites.length; i++) {</pre>
             graph.get(parent).add(child); // put the child into it's parent's list
inDegree.put(child, inDegree.get(child) + 1); // increment child's inDegree
          // c. Find all sources i.e., all vertices with 0 in-degrees Queue<Integer> sources = new LinkedList<>();
          for (Map.Entry<Integer, Integer> entry : inDegree.entrySet()) {
            if (entry.getValue() == 0)
    sources.add(entry.getKey()
                                                                                                                 SAVE
                                                                                                                                RESET
                                                                                                                                            03
                                                                                                                                      Close
                                                                                                                                      1.558s
Tasks execution possible: true
Tasks execution possible: false
Tasks execution possible: true
```

Time complexity

In step 'd', each task can become a source only once and each edge (prerequisite) will be accessed and removed once. Therefore, the time complexity of the above algorithm will be O(V+E), where 'V' is the total number of tasks and 'E' is the total number of prerequisites.

Space complexity

The space complexity will be O(V+E),), since we are storing all of the prerequisites for each task in an adjacency list.

Similar Problems

Course Schedule: There are 'N' courses, labeled from '0' to 'N-1'. Each course can have some prerequisite courses which need to be completed before it can be taken. Given the number of courses and a list of prerequisite pairs, find if it is possible for a student to take all the courses.

Solution: This problem is exactly similar to our parent problem. In this problem, we have courses instead of tasks.

