

Grokking the Coding Interview: Patterns for Coding Questions

93% completed



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) ▴

- Introduction
- Topological Sort (medium)
- Tasks Scheduling (medium)
- Tasks Scheduling Order (medium)**
- All Tasks Scheduling Orders (hard)
- Alien Dictionary (hard)

Tasks Scheduling Order (medium)

We'll cover the following ▴

- Problem Statement
- Try it yourself
- Solution
 - Code
 - Time complexity
 - Space complexity
- Similar Problems

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, write a method to find the ordering of tasks we should pick to finish all tasks.

Example 1:

```
Input: Tasks=3, Prerequisites=[0, 1], [1, 2]
Output: [0, 1, 2]
Explanation: To execute task '1', task '0' needs to finish first. Similarly, task '1' needs to finish before '2' can be scheduled. A possible scheduling of tasks is: [0, 1, 2]
```

Example 2:

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

Example 3:

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

Try it yourself

Try solving this question here:

JavaPython3JS C++

```
1 const find_order = function(tasks, prerequisites) {
2   sortedOrder = [];
3   // TODO: Write your code here
4   return sortedOrder;
5 };
6
7
8 console.log('Is scheduling possible: ${find_order(3, [[0, 1], [1, 2]])}')
9 console.log('Is scheduling possible: ${find_order(3, [[0, 1], [1, 2], [2, 0]])}')
10 console.log('Is scheduling possible: ${find_order(6, [[2, 5], [0, 5], [0, 4], [1, 4], [3, 2], [1, 3]])}')
11
```

RUNSAVERESET↺

Solution

This problem is similar to [Tasks Scheduling](#), the only difference being that we need to find the best ordering of tasks so that it is possible to schedule them all.

Code

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

JavaPython3C++JS

```
1 const Deque = require('./collections/deque'); //http://www.collectionsjs.com
2
3 function find_order(tasks, prerequisites) {
4   sortedOrder = [];
5   if (tasks <= 0) {
6     return sortedOrder;
7   }
8
9   // a. Initialize the graph
10  const inDegree = Array(tasks).fill(0); // count of incoming edges
11  const graph = Array(tasks).fill(0).map(() => Array()); // adjacency list graph
12
13  // b. Build the graph
```

Miscellaneous

- Kth Smallest Number (hard)

Conclusions

- Where to Go from Here

Mark Course as Completed

14prerequisites.forEach((prerequisite) => {
15 let parent = prerequisite[0],
16 child = prerequisite[1];
17 graph[parent].push(child); // put the child into it's parent's list
18 inDegree[child]++; // increment child's inDegree
19 });
20
21
22 // c. Find all sources i.e., all vertices with 0 in-degrees
23 const sources = new Deque();
24 for (i = 0; i < inDegree.length; i++) {
25 if (inDegree[i] === 0) {
26 sources.push(i);
27 }
28 }
}

RUN

SAVE

RESET

Close

4.568s

Output
Is scheduling possible: 0,1,2
Is scheduling possible:
Is scheduling possible: 0,1,5,4,3,2

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 has some prerequisite courses which need to be completed before it can be taken. Given the number of courses and a list of prerequisite pairs, write a method to find the best ordering of the courses that a student can take in order to finish all courses.

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

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. [See how](#)

← Back

Next →

Tasks Scheduling (medium)

All Tasks Scheduling Orders (hard)

✓ Mark as Completed

Report an Issue

Ask a Question