



## 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:



```
-- i y a iono
   def find_order(tasks, prerequisites):
 1
 2
      sortedOrder = []
 3
      # TODO: Write your code here
 4
      return sortedOrder
 6
 7
   def main():
 8
      print("Is scheduling possible: " + str(find_order(3, [[0, 1], [1, 2]])))
      print("Is scheduling possible: " +
 9
10
             str(find_order(3, [[0, 1], [1, 2], [2, 0]])))
11
      print("Is scheduling possible: " +
12
             str(find_order(6, [[2, 5], [0, 5], [0, 4], [1, 4], [3, 2], [1, 3]])))
13
14
15 main()
16
                                                                                            \leftarrow
                                                                                                  []
\triangleright
                                                                                      a
```

### Solution #

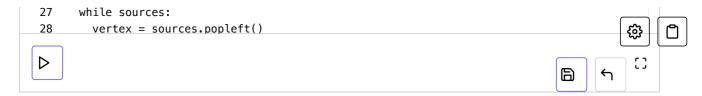
This problem is similar to Tasks Scheduling

(https://www.educative.io/collection/page/5668639101419520/5671464854355968/559002156426 8544/), 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):

```
🦆 Python3
                                     JS JS
Java
                         ③ C++
    from collections import deque
 3
 4
    def find_order(tasks, prerequisites):
 5
      sortedOrder = []
 6
      if tasks <= 0:
 7
         return sortedOrder
 8
 9
      # a. Initialize the graph
10
      inDegree = {i: 0 for i in range(tasks)} # count of incoming edges
11
      graph = {i: [] for i in range(tasks)} # adjacency list graph
12
13
      # b. Build the graph
14
      for prerequisite in prerequisites:
15
        parent, child = prerequisite[0], prerequisite[1]
16
        graph[parent].append(child) # put the child into it's parent's list
17
        inDegree[child] += 1 # increment child's inDegree
18
19
      # c. Find all sources i.e., all vertices with 0 in-degrees
20
      sources = deque()
21
      for key in inDegree:
22
        if inDegree[key] == 0:
23
          sources.append(key)
24
25
      # d. For each source, add it to the sortedOrder and subtract one from all of its childre
26
      # if a child's in-degree becomes zero, add it to the sources queue
```



#### 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.

