

All Tasks Scheduling Orders (hard)

We'll cover the following ^

- Problem Statement
- Try it yourself
- Solution
 - Code
 - Time and Space Complexity

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 print all possible ordering of tasks meeting all prerequisites.

Example 1:

```
Input: Tasks=3, Prerequisites=[0, 1], [1, 2]
Output: [0, 1, 2]
Explanation: There is only possible ordering of the tasks.
```

Example 2:

```
Input: Tasks=4, Prerequisites=[3, 2], [3, 0], [2, 0], [2, 1]
Output:
1) [3, 2, 0, 1]
2) [3, 2, 1, 0]
Explanation: There are two possible orderings of the tasks meeting all prerequisites.
```

Example 3:



Input: Tasks=6, Prerequisites=[2, 5], [0, 5], [0, 4], [1, 4], [3, 2], [1, 3]

Output:

- 1) [0, 1, 4, 3, 2, 5]
- 2) [0, 1, 3, 4, 2, 5]
- 3) [0, 1, 3, 2, 4, 5]
- 4) [0, 1, 3, 2, 5, 4]
- 5) [1, 0, 3, 4, 2, 5]
- 6) [1, 0, 3, 2, 4, 5]
- 7) [1, 0, 3, 2, 5, 4]
- 8) [1, 0, 4, 3, 2, 5]
- 9) [1, 3, 0, 2, 4, 5]
- 10) [1, 3, 0, 2, 5, 4]
- 11) [1, 3, 0, 4, 2, 5]
- 12) [1, 3, 2, 0, 5, 4]
- 13) [1, 3, 2, 0, 4, 5]

Try it yourself

Try solving this question here:

Java

Python3

JS

C++

```
1 def print_orders(tasks, prerequisites):
2     # TODO: Write your code here
3     print()
4
5
6
7 def main():
8     print("Task Orders: ")
9     print_orders(3, [[0, 1], [1, 2]])
10
11     print("Task Orders: ")
12     print_orders(4, [[3, 2], [3, 0], [2, 0], [2, 1]])
13
14     print("Task Orders: ")
15     print_orders(6, [[2, 5], [0, 5], [0, 4], [1, 4], [3, 2], [1, 3]])
16
17
18 main()
19
```

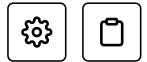
Solution

This problem is similar to Tasks Scheduling Order

(<https://www.educative.io/collection/page/5668639101419520/5671464854355968/5066018374287360/>), the only difference is that we need to find all the topological orderings of the tasks.

At any stage, if we have more than one source available and since we can choose any source, therefore, in this case, we will have multiple orderings of the tasks. We can use a recursive

approach with **Backtracking** to consider all sources at any step.



Code #

Here is what our algorithm will look like:

Java Python3 C++ JS

```
1 from collections import deque
2
3
4 def print_orders(tasks, prerequisites):
5     sortedOrder = []
6     if tasks <= 0:
7         return False
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    print_all_topological_sorts(graph, inDegree, sources, sortedOrder)
26
27
28 def print_all_topological_sorts(graph, inDegree, sources, sortedOrder):
```

Time and Space Complexity #

If we don't have any prerequisites, all combinations of the tasks can represent a topological ordering. As we know, that there can be $N!$ combinations for 'N' numbers, therefore the time and space complexity of our algorithm will be $O(V! * E)$ where 'V' is the total number of tasks and 'E' is the total prerequisites. We need the 'E' part because in each recursive call, at max, we remove (and add back) all the edges.

← Back

Next →

Tasks Scheduling Order (medium)

Alien Dictionary (hard)

✓ Completed

Report an Issue

Ask a Question

(https://discuss.educative.io/tag/all-tasks-scheduling-orders-hard__pattern-topological-sort-graph__grokking-the-coding-interview-patterns-for-coding-questions)

