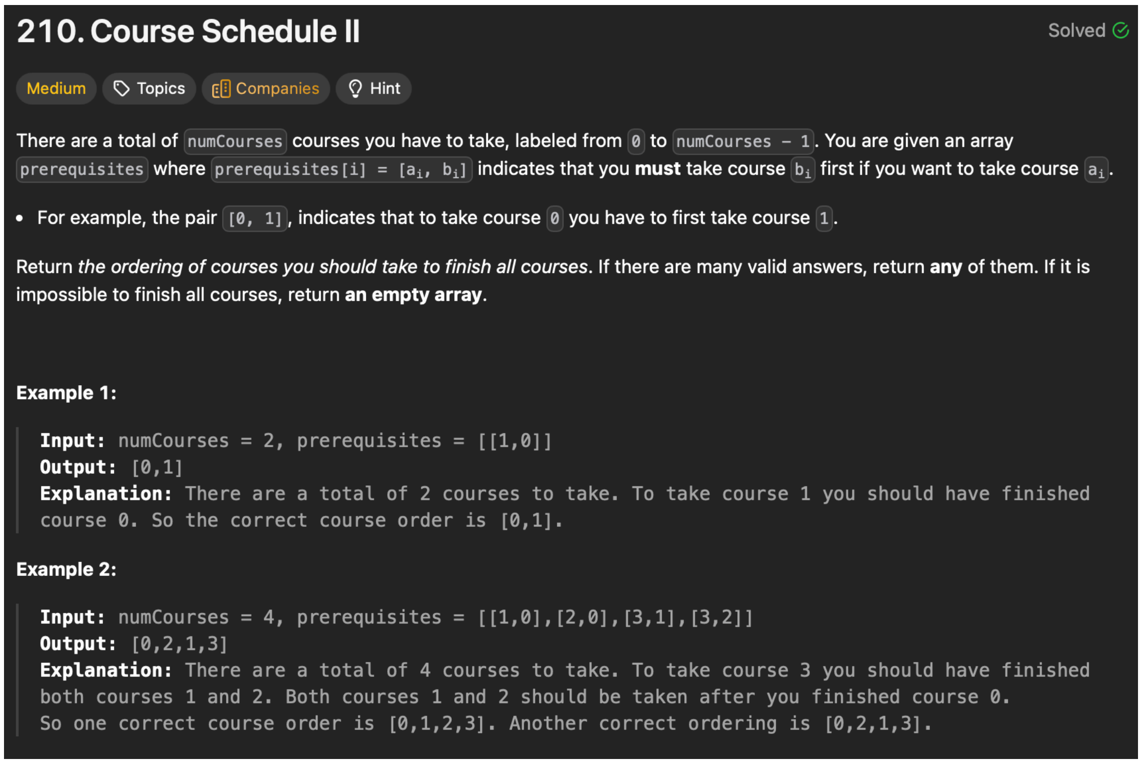


210. Course Schedule II

Solved

Medium | Topics | Companies | Hint

There are a total of `numCourses` courses you have to take, labeled from `0` to `numCourses - 1`. You are given an array `prerequisites` where `prerequisites[i] = [a_i, b_i]` indicates that you **must** take course `b_i` first if you want to take course `a_i`.

- For example, the pair `[0, 1]`, indicates that to take course `0` you have to first take course `1`.

Return *the ordering of courses you should take to finish all courses*. If there are many valid answers, return **any** of them. If it is impossible to finish all courses, return **an empty array**.

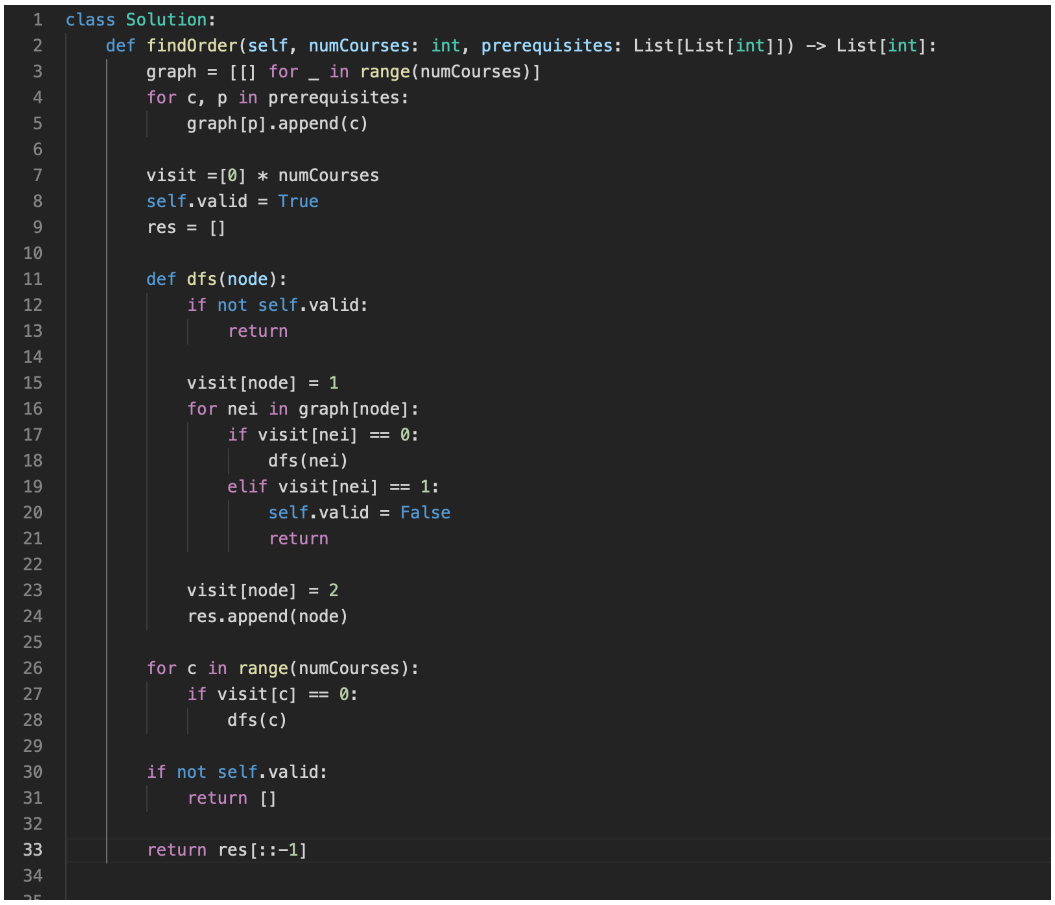
Example 1:

```
Input: numCourses = 2, prerequisites = [[1,0]]
Output: [0,1]
Explanation: There are a total of 2 courses to take. To take course 1 you should have finished
course 0. So the correct course order is [0,1].
```

Example 2:

```
Input: numCourses = 4, prerequisites = [[1,0],[2,0],[3,1],[3,2]]
Output: [0,2,1,3]
Explanation: There are a total of 4 courses to take. To take course 3 you should have finished
both courses 1 and 2. Both courses 1 and 2 should be taken after you finished course 0.
So one correct course order is [0,1,2,3]. Another correct ordering is [0,2,1,3].
```

DFS



```python
class Solution:
    def findOrder(self, numCourses: int, prerequisites: List[List[int]]) -> List[int]:
        graph = [[] for _ in range(numCourses)]
        for c, p in prerequisites:
            graph[p].append(c)

        visit =[0] * numCourses
        self.valid = True
        res = []

        def dfs(node):
            if not self.valid:
                return

            visit[node] = 1
            for nei in graph[node]:
                if visit[nei] == 0:
                    dfs(nei)
                elif visit[nei] == 1:
                    self.valid = False
                    return

            visit[node] = 2
            res.append(node)

        for c in range(numCourses):
            if visit[c] == 0:
                dfs(c)

        if not self.valid:
            return []

        return res[::-1]

```

visit == 0 -> unvisited; 1->visiting; 2->visited

if visit[x] == 1 and nei of x is also == 1, then it's a cycle, which will fail