```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

# O(j + d) time | O(j + d) space
def topologicalSort(jobs, deps):
    jobGraph = createJobGraph(jobs, deps)
    return getOrderedJobs(jobGraph)


def createJobGraph(jobs, deps):
    graph = JobGraph(jobs)
    for prereq, job in deps:
        graph.addPrereq(job, prereq)
    return graph


def getOrderedJobs(graph):
    orderedJobs = []
    nodes = graph.nodes
    while len(nodes):
        node = nodes.pop()
        containsCycle = depthFirstTraverse(node, orderedJobs)
        if containsCycle:
            return []
    return orderedJobs


def depthFirstTraverse(node, orderedJobs):
    if node.visited:
        return False
    if node.visiting:
        return True
    node.visiting = True
    for prereqNode in node.prereqs:
        containsCycle = depthFirstTraverse(prereqNode, orderedJobs)
        if containsCycle:
            return True
    node.visited = True
    node.visiting = False
    orderedJobs.append(node.job)
    return False


class JobGraph:
    def __init__(self, jobs):
        self.nodes = []
        self.graph = {}
        for job in jobs:
            self.addNode(job)

    def addPrereq(self, job, prereq):
        jobNode = self.getNode(job)
        prereqNode = self.getNode(prereq)
        jobNode.prereqs.append(prereqNode)

    def addNode(self, job):
        self.graph[job] = JobNode(job)
        self.nodes.append(self.graph[job])

    def getNode(self, job):
        if job not in self.graph:
            self.addNode(job)
        return self.graph[job]


class JobNode:
    def __init__(self, job):
        self.job = job
        self.prereqs = []
        self.visited = False
        self.visiting = False

```