```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 job, dep in deps:
        graph.addDep(job, dep)
    return graph


def getOrderedJobs(graph):
    orderedJobs = []
    nodesWithNoPrereqs = list(filter(lambda node: node.numOfPrereqs == 0, graph.nodes))
    while len(nodesWithNoPrereqs):
        node = nodesWithNoPrereqs.pop()
        orderedJobs.append(node.job)
        removeDeps(node, nodesWithNoPrereqs)
    graphHasEdges = any(node.numOfPrereqs for node in graph.nodes)
    return [] if graphHasEdges else orderedJobs


def removeDeps(node, nodesWithNoPrereqs):
    while len(node.deps):
        dep = node.deps.pop()
        dep.numOfPrereqs -= 1
        if dep.numOfPrereqs == 0:
            nodesWithNoPrereqs.append(dep)


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

    def addDep(self, job, dep):
        jobNode = self.getNode(job)
        depNode = self.getNode(dep)
        jobNode.deps.append(depNode)
        depNode.numOfPrereqs += 1

    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.deps = []
        self.numOfPrereqs = 0

```