```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

type Dep struct {
  Prereq int
  Job    int
}

// O(j + d) time | O(j + d) space
func TopologicalSort(jobs []int, deps []Dep) []int {
  jobGraph := createJobGraph(jobs, deps)
  return getOrderedJobs(jobGraph)
}

func createJobGraph(jobs []int, deps []Dep) *JobGraph {
  graph := NewJobGraph(jobs)
  for _, dep := range deps {
    graph.AddDep(dep.Prereq, dep.Job)
  }
  return graph
}

func getOrderedJobs(graph *JobGraph) []int {
  orderedJobs := []int{}
  nodesWithNoPrereqs := []*JobNode{}
  for _, node := range graph.Nodes {
    if node.NumOfPrereqs == 0 {
      nodesWithNoPrereqs = append(nodesWithNoPrereqs, node)
    }
  }
  for len(nodesWithNoPrereqs) > 0 {
    node := nodesWithNoPrereqs[len(nodesWithNoPrereqs)-1]
    nodesWithNoPrereqs = nodesWithNoPrereqs[:len(nodesWithNoPrereqs)-1]
    orderedJobs = append(orderedJobs, node.Job)
    removeDeps(node, &nodesWithNoPrereqs)
  }
  for _, node := range graph.Nodes {
    if node.NumOfPrereqs > 0 {
      return []int{}
    }
  }
  return orderedJobs
}

func removeDeps(node *JobNode, nodesWithNoPrereqs *[]*JobNode) {
  for len(node.Deps) > 0 {
    dep := node.Deps[len(node.Deps)-1]
    node.Deps = node.Deps[:len(node.Deps)-1]
    dep.NumOfPrereqs--
    if dep.NumOfPrereqs == 0 {
      *nodesWithNoPrereqs = append(*nodesWithNoPrereqs, dep)
    }
  }
}

type JobGraph struct {
  Nodes []*JobNode
  Graph map[int]*JobNode
}

func NewJobGraph(jobs []int) *JobGraph {
  g := &JobGraph{
    Graph: map[int]*JobNode{},
  }
  for _, job := range jobs {
    g.AddNode(job)
  }
  return g
}

func (g *JobGraph) AddDep(job, dep int) {
  jobNode, depNode := g.GetNode(job), g.GetNode(dep)
  jobNode.Deps = append(jobNode.Deps, depNode)
  depNode.NumOfPrereqs++
}

func (g *JobGraph) AddNode(job int) {
  g.Graph[job] = &JobNode{Job: job}
  g.Nodes = append(g.Nodes, g.Graph[job])
}

func (g *JobGraph) GetNode(job int) *JobNode {
  if _, found := g.Graph[job]; !found {
    g.AddNode(job)
  }
```