```java
    return getOrderedJobs(jobGraph);
  }

  public static JobGraph createJobGraph(List<Integer> jobs, List<Integer[]> deps) {
    JobGraph graph = new JobGraph(jobs);
    for (Integer[] dep : deps) {
      graph.addDep(dep[0], dep[1]);
    }
    return graph;
  }

  public static List<Integer> getOrderedJobs(JobGraph graph) {
    List<Integer> orderedJobs = new ArrayList<Integer>();
    List<JobNode> nodesWithNoPrereqs = new ArrayList<JobNode>();
    for (JobNode node : graph.nodes) {
      if (node.numOfPrereqs == 0) {
        nodesWithNoPrereqs.add(node);
      }
    }
    while (nodesWithNoPrereqs.size() > 0) {
      JobNode node = nodesWithNoPrereqs.get(nodesWithNoPrereqs.size() - 1);
      nodesWithNoPrereqs.remove(nodesWithNoPrereqs.size() - 1);
      orderedJobs.add(node.job);
      removeDeps(node, nodesWithNoPrereqs);
    }
    boolean graphHasEdges = false;
    for (JobNode node : graph.nodes) {
      if (node.numOfPrereqs > 0) {
        graphHasEdges = true;
      }
    }
    return graphHasEdges ? new ArrayList<Integer>() : orderedJobs;
  }

  public static void removeDeps(JobNode node, List<JobNode> nodesWithNoPrereqs) {
    while (node.deps.size() > 0) {
      JobNode dep = node.deps.get(node.deps.size() - 1);
      node.deps.remove(node.deps.size() - 1);
      dep.numOfPrereqs--;
      if (dep.numOfPrereqs == 0) nodesWithNoPrereqs.add(dep);
    }
  }

  static class JobGraph {
    public List<JobNode> nodes;
    public Map<Integer, JobNode> graph;

    public JobGraph(List<Integer> jobs) {
      nodes = new ArrayList<JobNode>();
      graph = new HashMap<Integer, JobNode>();
      for (Integer job : jobs) {
        addNode(job);
      }
    }

    public void addDep(Integer job, Integer dep) {
      JobNode jobNode = getNode(job);
      JobNode depNode = getNode(dep);
      jobNode.deps.add(depNode);
      depNode.numOfPrereqs++;
    }

    public void addNode(Integer job) {
      graph.put(job, new JobNode(job));
      nodes.add(graph.get(job));
    }

    public JobNode getNode(Integer job) {
      if (!graph.containsKey(job)) addNode(job);
      return graph.get(job);
    }
  }

  static class JobNode {
    public Integer job;
    public List<JobNode> deps;
    public Integer numOfPrereqs;

    public JobNode(Integer job) {
      this.job = job;
      deps = new ArrayList<JobNode>();
      numOfPrereqs = 0;
    }
  }
}
```