```cpp

  JobGraph(vector<int> jobs);
  void addPrereq(int job, int prereq);
  void addNode(int job);
  JobNode *getNode(int job);
};

JobGraph *createJobGraph(vector<int> jobs, vector<vector<int>> deps);
vector<int> getOrderedJobs(JobGraph *graph);
bool depthFirstTraverse(JobNode *node, vector<int> *orderedJobs);

// O(j + d) time | O(j + d) space
vector<int> topologicalSort(vector<int> jobs, vector<vector<int>> deps) {
  JobGraph *jobGraph = createJobGraph(jobs, deps);
  return getOrderedJobs(jobGraph);
}

JobGraph *createJobGraph(vector<int> jobs, vector<vector<int>> deps) {
  JobGraph *graph = new JobGraph(jobs);
  for (vector<int> dep : deps) {
    graph->addPrereq(dep[1], dep[0]);
  }
  return graph;
}

vector<int> getOrderedJobs(JobGraph *graph) {
  vector<int> orderedJobs = {};
  vector<JobNode *> nodes = graph->nodes;
  while (nodes.size()) {
    JobNode *node = nodes.back();
    nodes.pop_back();
    bool containsCycle = depthFirstTraverse(node, &orderedJobs);
    if (containsCycle)
      return {};
  }
  return orderedJobs;
}

bool depthFirstTraverse(JobNode *node, vector<int> *orderedJobs) {
  if (node->visited)
    return false;
  if (node->visiting)
    return true;
  node->visiting = true;
  for (JobNode *prereqNode : node->prereqs) {
    bool containsCycle = depthFirstTraverse(prereqNode, orderedJobs);
    if (containsCycle)
      return true;
  }
  node->visited = true;
  node->visiting = false;
  orderedJobs->push_back(node->job);
  return false;
}

JobGraph::JobGraph(vector<int> jobs) {
  nodes = {};
  for (int job : jobs) {
    addNode(job);
  }
}

void JobGraph::addPrereq(int job, int prereq) {
  JobNode *jobNode = getNode(job);
  JobNode *prereqNode = getNode(prereq);
  jobNode->prereqs.push_back(prereqNode);
}

void JobGraph::addNode(int job) {
  graph[job] = new JobNode(job);
  nodes.push_back(graph[job]);
}

JobNode *JobGraph::getNode(int job) {
  if (graph.find(job) == graph.end())
    addNode(job);
  return graph[job];
}

JobNode::JobNode(int job) {
  this->job = job;
  prereqs = {};
  visited = false;
  visiting = false;
}

```