AlgoExpert Quad Layout C# 12px Sublime Monokai 00:00:00

Prompt Scratchpad Our Solution(s) Video Explanation Run Code

Solution 1 Solution 2

```
    return getOrderedJobs(jobGraph);
  }

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

  public static List<int> getOrderedJobs(JobGraph graph) {
    List<int> orderedJobs = new List<int>();
    List<JobNode> nodesWithNoPrereqs = new List<JobNode>();
    foreach (JobNode node in graph.nodes) {
      if (node.numOfPrereqs == 0) {
        nodesWithNoPrereqs.Add(node);
      }
    }
    while (nodesWithNoPrereqs.Count > 0) {
      JobNode node = nodesWithNoPrereqs[nodesWithNoPrereqs.Count - 1];
      nodesWithNoPrereqs.RemoveAt(nodesWithNoPrereqs.Count - 1);
      orderedJobs.Add(node.job);
      removeDeps(node, nodesWithNoPrereqs);
    }
    bool graphHasEdges = false;
    foreach (JobNode node in graph.nodes) {
      if (node.numOfPrereqs > 0) {
        graphHasEdges = true;
      }
    }
    return graphHasEdges ? new List<int>() : orderedJobs;
  }

  public static void removeDeps(JobNode node, List<JobNode> nodesWithNoPrereqs) {
    while (node.deps.Count > 0) {
      JobNode dep = node.deps[node.deps.Count - 1];
      node.deps.RemoveAt(node.deps.Count - 1);
      dep.numOfPrereqs--;
      if (dep.numOfPrereqs == 0) nodesWithNoPrereqs.Add(dep);
    }
  }

  public class JobGraph {
    public List<JobNode> nodes;
    public Dictionary<int, JobNode> graph;

    public JobGraph(List<int> jobs) {
      nodes = new List<JobNode>();
      graph = new Dictionary<int, JobNode>();
      foreach (int job in jobs) {
        addNode(job);
      }
    }

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

    public void addNode(int job) {
      graph.Add(job, new JobNode(job));
      nodes.Add(graph[job]);
    }

    public JobNode getNode(int job) {
      if (!graph.ContainsKey(job)) addNode(job);
      return graph[job];
    }
  }

  public class JobNode {
    public int job;
    public List<JobNode> deps;
    public int numOfPrereqs;

    public JobNode(int job) {
      this.job = job;
      deps = new List<JobNode>();
      numOfPrereqs = 0;
    }
  }
}

```