Prompt Scratchpad Our Solution(s) Video Explanation Run Code

00:00:00

```
Solution 1
               Solution 2
        return getOrderedJobs(jobGraph);
10
11
      public static JobGraph createJobGraph(List<Integer> jobs, List<Integer[]> deps) {
12
13
        JobGraph graph = new JobGraph(jobs);
14
        for (Integer[] dep : deps) {
15
          graph.addDep(dep[0], dep[1]);
16
17
        return graph;
18
19
20
      public static List<Integer> getOrderedJobs(JobGraph graph) {
21
        List<Integer> orderedJobs = new ArrayList<Integer>();
22
        List<JobNode> nodesWithNoPrereqs = new ArrayList<JobNode>();
23
        for (JobNode node : graph.nodes) {
24
          if (node.numOfPrereqs == 0) {
25
            nodesWithNoPrereqs.add(node);
26
27
28
        while (nodesWithNoPrereqs.size() > 0) {
          \label{local_problem} \mbox{\tt JobNode node = nodesWithNoPrereqs.get(nodesWithNoPrereqs.size() - 1);}
29
30
          nodesWithNoPrereqs.remove(nodesWithNoPrereqs.size() - 1);
31
          orderedJobs.add(node.job);
32
          removeDeps(node, nodesWithNoPrereqs);
33
34
        boolean graphHasEdges = false;
35
        for (JobNode node : graph.nodes) {
          if (node.numOfPrereqs > 0) {
36
37
            graphHasEdges = true;
38
39
40
        return graphHasEdges ? new ArrayList<Integer>() : orderedJobs;
41
42
      public static void removeDeps(JobNode node, List<JobNode> nodesWithNoPrereqs) {
43
44
        while (node.deps.size() > 0) {
45
          JobNode dep = node.deps.get(node.deps.size() - 1);
          node.deps.remove(node.deps.size() - 1);
46
47
          dep.numOfPrereqs--;
48
          if (dep.numOfPrereqs == 0) nodesWithNoPrereqs.add(dep);
49
50
51
52
      static class JobGraph {
53
        public List<JobNode> nodes;
54
        public Map<Integer, JobNode> graph;
55
        public JobGraph(List<Integer> jobs) {
56
          nodes = new ArrayList<JobNode>();
57
58
          graph = new HashMap<Integer, JobNode>();
59
          for (Integer job : jobs) {
60
            addNode(job);
61
62
63
        public void addDep(Integer job, Integer dep) {
64
65
          JobNode jobNode = getNode(job);
          JobNode depNode = getNode(dep);
66
67
          jobNode.deps.add(depNode);
68
          depNode.numOfPrereqs++;
69
70
        public void addNode(Integer job) {
71
72
          graph.put(job, new JobNode(job));
73
          nodes.add(graph.get(job));
74
75
76
        public JobNode getNode(Integer job) {
77
          if (!graph.containsKey(job)) addNode(job);
78
          return graph.get(job);
79
80
81
      static class JobNode {
82
83
        public Integer job;
84
        public List<JobNode> deps;
85
        public Integer numOfPrereqs;
86
87
        public JobNode(Integer job) {
88
          this.job = job;
          deps = new ArravList<JobNode>():
         numOfPrereqs = 0;
90
91
92
    }
93 }
94
```