

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 // O(j + d) time | O(j + d) space
4 function topologicalSort(jobs, deps) {
5   const jobGraph = createJobGraph(jobs, deps);
6   return getOrderedJobs(jobGraph);
7 }
8
9 function createJobGraph(jobs, deps) {
10  const graph = new JobGraph(jobs);
11  for (const [job, dep] of deps) {
12    graph.addDep(job, dep);
13  }
14  return graph;
15 }
16
17 function getOrderedJobs(graph) {
18  const orderedJobs = [];
19  const nodesWithNoPrereqs = graph.nodes.filter(node => !node.numOfPre
20  while (nodesWithNoPrereqs.length) {
21    const node = nodesWithNoPrereqs.pop();
22    orderedJobs.push(node.job);
23    removeDeps(node, nodesWithNoPrereqs);
24  }
25  const graphHasEdges = graph.nodes.some(node => node.numOfPrereqs);
26  return graphHasEdges ? [] : orderedJobs;
27 }
28
29 function removeDeps(node, nodesWithNoPrereqs) {
30  while (node.deps.length) {
31    const dep = node.deps.pop();
32    dep.numOfPrereqs--;
33    if (!dep.numOfPrereqs) nodesWithNoPrereqs.push(dep);
```

Solution 1

Solution 2

Solution 3

```
1 function topologicalSort(jobs, deps) {
2   // Write your code here.
3 }
4
5 // Do not edit the line below.
6 exports.topologicalSort = topologicalSort;
7
```

Our Tests

Custom Output

Submit Code

1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.

2

3 // O(j + d) time | O(j + d) space

4 function topologicalSort(jobs, deps) {

5 const jobGraph = createJobGraph(jobs, deps);

6 return getOrderedJobs(jobGraph);

7 }

8

9 function createJobGraph(jobs, deps) {

10 const graph = new JobGraph(jobs);

11 for (const [job, dep] of deps) {

12 graph.addDep(job, dep);

13 }

14 return graph;

15 }

16

17 function getOrderedJobs(graph) {

18 const orderedJobs = [];

19 const nodesWithNoPrereqs = graph.nodes.filter(node => !node.numOfPre

20 while (nodesWithNoPrereqs.length) {

21 const node = nodesWithNoPrereqs.pop();

22 orderedJobs.push(node.job);

23 removeDeps(node, nodesWithNoPrereqs);

24 }

25 const graphHasEdges = graph.nodes.some(node => node.numOfPrereqs);

26 return graphHasEdges ? [] : orderedJobs;

27 }

28

29 function removeDeps(node, nodesWithNoPrereqs) {

30 while (node.deps.length) {

31 const dep = node.deps.pop();

32 dep.numOfPrereqs--;

33 if (!dep.numOfPrereqs) nodesWithNoPrereqs.push(dep);

34 }

35 }

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

```

18  # 20, 30,
19  # 20, 30,
20  # 20, 30,
21  # 20, 40,
22  # 20, 40,
23  # 20, 40,
24  # 20, 40,
25  # 20, 40,
26  # 20, 40,
27  # 20, 40,
28  # 20, 40,
29  # 20, 40,
30  # 20, 40,
31  # 20, 40,
32  # 20, 40,
33  # 20, 40,
34  # 20, 40,
35  # 20, 40,
36  # 20, 40,
37  # 20, 40,
38  # 20, 40,
39  # 20, 40,
40  # 20, 40,
41  # 20, 40,
42  # 20, 40,
43  # 20, 40,
44  # 20, 40,
45  # 20, 40,
46  # 20, 40,
47  # 20, 40,
48  # 20, 40,
49  # 20, 40,
50  # 20, 40,
51  # 20, 40,
52  # 20, 40,
53  # 20, 40,
54  # 20, 40,
55  # 20, 40,
56  # 20, 40,
57  # 20, 40,
58  # 20, 40,
59  # 20, 40,
60  # 20, 40,
61  # 20, 40,
62  # 20, 40,
63  # 20, 40,
64  # 20, 40,
65  # 20, 40,
66  # 20, 40,
67  # 20, 40,
68  # 20, 40,
69  # 20, 40,
70  # 20, 40,
71  # 20, 40,
72  # 20, 40,
73  # 20, 40,
74  # 20, 40,
75  # 20, 40,
76  # 20, 40,
77  # 20, 40,
78  # 20, 40,
79  # 20, 40,
80  # 20, 40,
81  # 20, 40,
82  # 20, 40,
83  # 20, 40,
84  # 20, 40,
85  # 20, 40,
86  # 20, 40,
87  # 20, 40,
88  # 20, 40,
89  # 20, 40,
90  # 20, 40,
91  # 20, 40,
92  # 20, 40,
93  # 20, 40,
94  # 20, 40,
95  # 20, 40,
96  # 20, 40,
97  # 20, 40,
98  # 20, 40,
99  # 20, 40,
100 # 20, 40,

```

Run or submit code when you're ready.