Our Solution(s) Run Code

Solution 1 Solution 2

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

using System.Collections.Generic;

public class Program {
  // O(j + d) time | O(j + d) space
  public static List<int> TopologicalSort(List<int> jobs, List<int[]>
    JobGraph jobGraph = createJobGraph(jobs, deps);
    return getOrderedJobs(jobGraph);
  }

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

  public static List<int> getOrderedJobs(JobGraph graph) {
    List<int> orderedJobs = new List<int>();
    List<JobNode> nodes = new List<JobNode>(graph.nodes);
    while (nodes.Count > 0) {
      JobNode node = nodes[nodes.Count - 1];
      nodes.RemoveAt(nodes.Count - 1);
      bool ContainsCycle = depthFirstTraverse(node, orderedJobs);
      if (ContainsCycle) return new List<int>();
    }
    return orderedJobs;
  }

  public static bool depthFirstTraverse(JobNode node, List<int> ordere
    if (node.visited) return false;
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```
using System.Collections.Generic;

public class Program {
  public static List<int> TopologicalSort(List<int> jobs, List<int[]>
    // Write your code here.
    return null;
  }
}

```

Our Tests

Custom Output Submit Code

```
    };
    int[,] depthMap = new int[,] {
      {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    };
    List<int>[] dups = new List<int>[1];
    FillDups(depthMap, dups);
    List<int> order = Program.TopologicalSort(jobs, dups);
    Write.Assert(true|IsTopologicalOrder(order, jobs, dups) == 1);
  }

  [Test]
  public void TestCase2() {
    List<int> jobs = new List<int>{
      1, 2, 3, 4, 5, 6, 7, 8
    };
```

Run or submit code when you're ready.