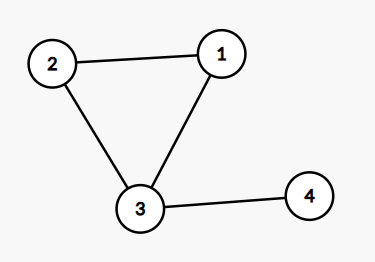
# Lab Exam Problems

1. Write code to solve the single source shortest path problem on a **DAG** using **DFS**.  
   Take both the **DAG** and the **source node** as input and output the **distance of each node**.  
   (You can choose any graph representation or input format of your choice)
2. Write code to solve the following grid traversal problem. **You don’t need to print the path.**  <https://cses.fi/problemset/task/1194>
3. Write code to solve **cycle detection** in a **directed graph** using **BFS.**
4. We have seen cycle detection in a **directed** graph .   
   Now we are interested in detecting cycles in an **undirected** graph using **DFS**. A cycle in an undirected graph has **at least** **3** nodes in it. For example, the following graph has a cycle consisting nodes **1, 2 & 3.   
   **You can find the pseudocode for detecting cycles in an **undirected graph** using **DFS** in this link: <https://ideone.com/ZdPwm3>

Now, write code to solve the problem: <https://cses.fi/problemset/task/1669>

**You don’t need to print the path.**

1. Write code to solve the problem <https://cses.fi/problemset/task/1669> again using **BFS.** Can you come up with your own algorithm?
2. Write code to solve the topological sorting problem <https://cses.fi/problemset/task/1679/>using **BFS**.   
   You can find the pseudocode for implementing topsort using **BFS** in this link: <https://ideone.com/6L967A>  
     
   Can you give an intuitive description of why this algorithm works?
3. We solved the flood fill problem <https://cses.fi/problemset/task/1192> with the following code:  
   <https://github.com/phitronio/Algorithm-Batch1/blob/main/module%2009/flood_fill.cpp>  
     
   What is the time complexity of this code? Can you come up with something better?  
   **Hint:** maybe the **while(true)** loop for finding an **unvisited cell** is overkill?
4. Write code to solve the following problem: <https://cses.fi/problemset/task/1666>