



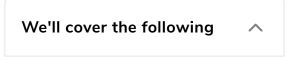






Solution Review: Check if Given Undirected Graph is a Tree or Not

This review provides a detailed analysis of the different ways to check if an undirected graph is a tree or not.



- Solution: Recursion stack
 - Time complexity

Solution: Recursion stack#

```
main.py

Graph.py

Stack.py

Queue.py

LinkedList.py

Node.py

1 from Graph import Graph
2 # We only need Graph for this Question!
3
4
5 def is_tree(g):
6 # All vertices unvisited
7 visited = [False] * g.vertices
```

```
# Check cycle using recursion stack
 9
                                                             []
10
        # Also mark nodes visited to check connectivity
        if check_cycle(g, 0, visited, -1):
11
            return False
12
13
        # Check if all nodes we visited from the source (graph is connect
14
15
        for i in range(len(visited)):
16
            # Graph is not connected
17
            if not visited[i]:
                return False
18
19
        # Not cycle and connected graph
20
        return True
21
22
23
    def check_cycle(g, node, visited, parent):
        # Mark node as visited
24
25
        visited[node] = True
26
        # Pick adjacent node and run recursive DFS
27
        adjacent = g.array[node].head_node
28
```

The logic for this problem is the same as Challenge 3 where you have to detect a cycle in the graph. We make a stack (not to be confused with the stack data structure) of vertices in <code>check_cycle()</code>. This stack grows recursively (line 31). The only difference is that we keep track of the <code>parent</code> vertex since a backward link to the parent does not count as a cycle (undirected graph). If a cycle is found in the graph, <code>check_cycle</code> will return <code>True</code>.

At the end of our recursion, two things must be true if the graph is a tree:

- All elements of visited must be true
- check_cycle should return False

Whenever these two conditions are true, our graph is a tree!



Time complexity#







The graph is traversed in both functions. Hence, the time complexity is O(V +*E*).

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. See how ①





Challenge 7: Check if a Given Undirect...



Challenge 8: Find the Shortest Path B...



✓ Mark as Completed



Report an Issue

