```
        return count
```

We just need to add few condition checks in order to extend above solution for Graph Valid Tree.

1. Check if number of components(`count`) is 1 i.e. all the nodes are connected with each other.
2. Check if the number of edges are one less than number of vertices. (Essential condition for a graph to be a Tree.)

DFS Solution for Graph Valid Tree(#261):

```
class Solution:
    def validTree(self, n: int, edges: List[List[int]]) -> bool:

        def dfs(n,graph,visited):
            if visited[n]:
                return
            visited[n] = 1
            for x in graph[n]:
                dfs(x,graph,visited)

        visited, count = [0]*n, 0
        graph = {x:[] for x in range(n)}
        for x,y in edges:
            graph[x].append(y)
            graph[y].append(x)

        for i in range(n):
            if not visited[i]:
                dfs(i,graph,visited)
                count += 1

        """Additional Condition Check."""

        if count==1 and len(edges) < n:
            return True
        else:
            return False
```

python3 python dfs