```
from collections import defaultdict
class Graph:
    def __init__(self):
        self.graph = defaultdict(list)
    def add_edge(self, u, v):
        self.graph[u].append(v)
    def dls(self, node, goal, depth, current_depth=0):
        print(f"Visiting Node: {node}, Current Depth: {current_depth}")
        if node == goal:
            return True
        if current_depth >= depth:
            return False
        for neighbor in self.graph.get(node, []):
            if self.dls(neighbor, goal, depth, current_depth + 1):
                return True
        return False
g = Graph()
g.add_edge(0, 1)
g.add_edge(0, 2)
g.add_edge(1, 3)
g.add_edge(1, 4)
g.add_edge(2, 5)
g.add_edge(2, 6)
g.add_edge(3, 7)
g.add_edge(3, 8)
g.add_edge(4, 9)
g.add_edge(5, 10)
g.add_edge(6, 11)
g.add_edge(6, 12)
g.add_edge(7, 13)
g.add_edge(8, 14)
g.add_edge(9, 15)
g.add_edge(10, 16)
g.add_edge(10, 17)
start_node = 0
goal_node = 16
depth_limit = 4
if g.dls(start_node, goal_node, depth_limit):
print(f"Goal node {goal_node} found within depth limit {depth_limit}")
else:
print(f"Goal node {goal_node} NOT found within depth limit
{depth_limit}")
```