61. Minimum Time to Collect All Apples in a Tree

Given an undirected tree consisting of n vertices numbered from 0 to n-1, which has some apples in their vertices. You spend 1 second to walk over one edge of the tree. Return the minimum time in seconds you have to spend to collect all apples in the tree, starting at vertex 0 and coming back to this vertex.

The edges of the undirected tree are given in the array edges, where edges[i] = [ai, bi] means that exists an edge connecting the vertices ai and bi. Additionally, there is a boolean array hasApple, where hasApple[i] = true means that vertex i has an apple; otherwise, it does not have any apple.

Program:

```
def minTime(n, edges, hasApple):
    from collections import defaultdict

    # Create an adjacency list
    tree = defaultdict(list)
    for u, v in edges:
        tree[u].append(v)
        tree[v].append(u)

    def dfs(node, parent):
        time = 0
        for neighbor in tree[node]:
            if neighbor != parent:
                time += dfs(neighbor, node)

        if (time > 0 or hasApple[node]) and node != 0:
            time += 2
        return time

    return dfs(0, -1)

# Example usage
n = 7
edges = [[0, 1], [0, 2], [1, 4], [1, 5], [2, 3], [2, 6]]
```

hasApple = [False, False, True, False, True, True, False]

print(minTime(n, edges, hasApple)) # Output: 8

n = 7

edges = [[0, 1], [0, 2], [1, 4], [1, 5], [2, 3], [2, 6]]

hasApple = [False, False, True, False, False, True, False]

print(minTime(n, edges, hasApple)) # Output: 6

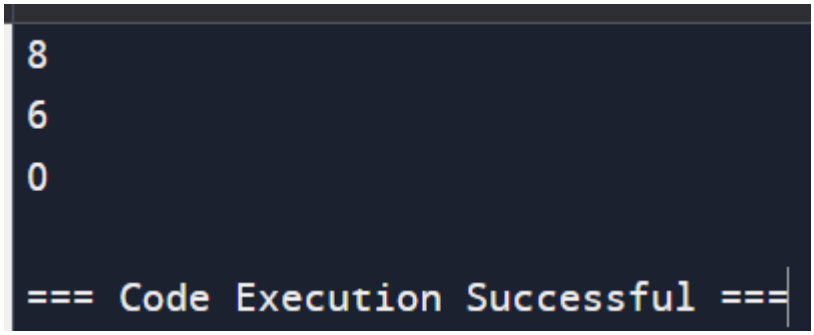
n = 7

edges = [[0, 1], [0, 2], [1, 4], [1, 5], [2, 3], [2, 6]]

hasApple = [False, False, False, False, False, False, False]

print(minTime(n, edges, hasApple)) # Output: 0

Output:



Time complexity:O(n)