Program 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.

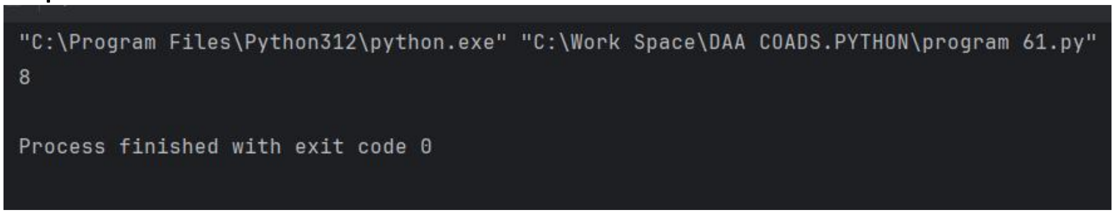
Example 1:

Input: n = 7, edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], hasApple = [false,false,true,false,true,true,false]
Output: 8
Explanation: The figure above represents the given tree where red vertices have an apple. One optimal path to collect all apples is shown by the green arrows.

Program:

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

    # Build the graph
    graph = defaultdict(list)
    for u, v in edges:
        graph[u].append(v)
        graph[v].append(u)

    def dfs(node, parent):
        total_time = 0

        # Traverse all children
        for child in graph[node]:
            if child != parent:
                time = dfs(child, node)
                # If the child has an apple or any of its descendants have apples
                if time > 0 or hasApple[child]:
                    total_time += time + 2

        return total_time

    # Perform DFS starting from node 0
    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
```

Output:



```
"C:\Program Files\Python312\python.exe" "C:\Work Space\DAA COADS.PYTHON\program 61.py"
8

Process finished with exit code 0
```

Time complexity:
O(n)