EXERCISE-60 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

PROGRAM;

```
def min_time_to_collect_apples(n, edges, hasApple):
    graph = {i: [] for i in range(n)}
    for edge in edges:
        graph[edge[0]].append(edge[1])
        graph[edge[1]].append(edge[0])

    visited = set()

    def dfs(node):
        if node in visited:
            return 0
        visited.add(node)
        time = 0
        for neighbor in graph[node]:
            time += dfs(neighbor)
        return time + (2 if (time > 0 or hasApple[node]) and node != 0 else 0)

    return dfs(0)

# Test case
n = 7
edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]]
hasApple = [False, False, True, False, True, True, False]
print(min_time_to_collect_apples(n, edges, hasApple))
```

OUTPUT;

```
====
8
```

TIME COMPLEXITY; O(V + E),