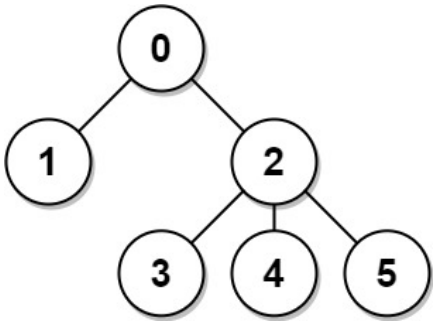
834. Sum of Distances in Tree

There is an undirected connected tree with n nodes labeled from 0 to n - 1 and n - 1 edges.

You are given the integer n and the array edges where edges[i] = [a_i, b_i] indicates that there is an edge between nodes a_i and b_i in the tree.

Return an array answer of length n where answer[i] is the sum of the distances between the i^{th} node in the tree and all other nodes.

Example 1:



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

Output: [8,12,6,10,10,10]

Explanation: The tree is shown above.

We can see that dist(0,1) + dist(0,2) + dist(0,3) + dist(0,4) + dist(0,5)

equals 1 + 1 + 2 + 2 + 2 = 8.

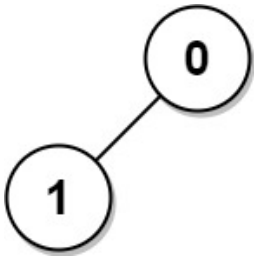
Hence, answer[0] = 8, and so on.

Example 2:

Input: n = 1, edges = []

Output: [0]

Example 3:



Input: n = 2, edges = [[1,0]]

Output: [1,1]

```
from collections import defaultdict
class Solution(object):
    def sumOfDistancesInTree(self, n, edges):
        tree = defaultdict(list)
        for u, v in edges:
            tree[u].append(v)
            tree[v].append(u)

        subtree_sizes = [0] * n
        dist_sum = [0] * n

        def dfs(node, parent):
            size = 1
            distance = 0
            for child in tree[node]:
                if child != parent:
                    dfs(child, node)
                    size += subtree_sizes[child]
                    distance += subtree_sizes[child] +
dist_sum[child]
            subtree_sizes[node] = size
            dist_sum[node] = distance

        dfs(0, -1)
        ans = [0] * n
        ans[0] = dist_sum[0]
        def dfs2(node, parent):
            for child in tree[node]:
                if child != parent:
                    ans[child] = ans[node] -
subtree_sizes[child] + n - subtree_sizes[child]
                    dfs2(child, node)

        dfs2(0, -1)

        return ans
```