Problem 3715: Sum of Perfect Square Ancestors

Problem Information

Difficulty: Hard
Acceptance Rate: 41.30%
Paid Only: No
Tags: Array, Hash Table, Math, Tree, Depth-First Search, Counting, Number Theory

Problem Description

You are given an integer `n` and an undirected tree rooted at node 0 with `n` nodes numbered from 0 to `n - 1`. This is represented by a 2D array `edges` of length `n - 1`, where `edges[i] = [ui, vi]` indicates an undirected edge between nodes `ui` and `vi`.

You are also given an integer array `nums`, where `nums[i]` is the positive integer assigned to node `i`.

Define a value `ti` as the number of **ancestors** of node `i` such that the product `nums[i] * nums[ancestor]` is a **perfect square**.

Return the sum of all `ti` values for all nodes `i` in range `[1, n - 1]`.

Note :

* In a rooted tree, the **ancestors** of node `i` are all nodes on the path from node `i` to the root node 0, **excluding** `i` itself.

Example 1:

Input: n = 3, edges = [[0,1],[1,2]], nums = [2,8,2]

Output: 3

Explanation:

`**i**` | **Ancestors** | `**nums[i] * nums[ancestor]**` | Square Check | `**t i**` ---|---|---|---|--- 1 | [0] | `nums[1] * nums[0] = 8 * 2 = 16` | 16 is a perfect square | 1 2 | [1, 0] | `nums[2] * nums[1] = 2 * 8 = 16` `nums[2] * nums[0] = 2 * 2 = 4` | Both 4 and 16 are perfect squares | 2 Thus, the total number of valid ancestor pairs across all non-root nodes is `1 + 2 = 3`.

Example 2:

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

Output: 1

Explanation:

`**i**` | **Ancestors** | `**nums[i] * nums[ancestor]**` | Square Check | `**t i**` ---|---|---|---|--- 1 | [0] | `nums[1] * nums[0] = 2 * 1 = 2` | 2 is **not** a perfect square | 0 2 | [0] | `nums[2] * nums[0] = 4 * 1 = 4` | 4 is a perfect square | 1 Thus, the total number of valid ancestor pairs across all non-root nodes is 1.

Example 3:

Input: n = 4, edges = [[0,1],[0,2],[1,3]], nums = [1,2,9,4]

Output: 2

Explanation:

`i` | **Ancestors** | `**nums[i] * nums[ancestor]**` | Square Check | `**t i**` ---|---|---|---|--- 1 | [0] | `nums[1] * nums[0] = 2 * 1 = 2` | 2 is **not** a perfect square | 0 2 | [0] | `nums[2] * nums[0] = 9 * 1 = 9` | 9 is a perfect square | 1 3 | [1, 0] | `nums[3] * nums[1] = 4 * 2 = 8` `nums[3] * nums[0] = 4 * 1 = 4` | Only 4 is a perfect square | 1 Thus, the total number of valid ancestor pairs across all non-root nodes is `0 + 1 + 1 = 2`.

Constraints:

* `1 <= n <= 105` * `edges.length == n - 1` * `edges[i] = [ui, vi]` * `0 <= ui, vi <= n - 1` * `nums.length == n` * `1 <= nums[i] <= 105` * The input is generated such that `edges` represents a valid tree.

Code Snippets

C++:

```
class Solution {
public:
long long sumOfAncestors(int n, vector<vector<int>>& edges, vector<int>&
nums) {

}
};
```

Java:

```
class Solution {
public long sumOfAncestors(int n, int[][] edges, int[] nums) {

}
}
```

Python3:

```
class Solution:
def sumOfAncestors(self, n: int, edges: List[List[int]], nums: List[int]) ->
int:
```