Problem 3372: Maximize the Number of Target Nodes After Connecting Trees I

Problem Information

Difficulty: Medium
Acceptance Rate: 69.53%
Paid Only: No
Tags: Tree, Depth-First Search, Breadth-First Search

Problem Description

There exist two **undirected** trees with `n` and `m` nodes, with **distinct** labels in ranges `[0, n - 1]` and `[0, m - 1]`, respectively.

You are given two 2D integer arrays `edges1` and `edges2` of lengths `n - 1` and `m - 1`, respectively, where `edges1[i] = [ai, bi]` indicates that there is an edge between nodes `ai` and `bi` in the first tree and `edges2[i] = [ui, vi]` indicates that there is an edge between nodes `ui` and `vi` in the second tree. You are also given an integer `k`.

Node `u` is **target** to node `v` if the number of edges on the path from `u` to `v` is less than or equal to `k`. **Note** that a node is _always_ **target** to itself.

Return an array of `n` integers `answer`, where `answer[i]` is the **maximum** possible number of nodes **target** to node `i` of the first tree if you have to connect one node from the first tree to another node in the second tree.

Note that queries are independent from each other. That is, for every query you will remove the added edge before proceeding to the next query.

Example 1:

Input: edges1 = [[0,1],[0,2],[2,3],[2,4]], edges2 = [[0,1],[0,2],[0,3],[2,7],[1,4],[4,5],[4,6]], k = 2

Output: [9,7,9,8,8]

Explanation:

* For `i = 0`, connect node 0 from the first tree to node 0 from the second tree. * For `i = 1`, connect node 1 from the first tree to node 0 from the second tree. * For `i = 2`, connect node 2 from the first tree to node 4 from the second tree. * For `i = 3`, connect node 3 from the first tree to node 4 from the second tree. * For `i = 4`, connect node 4 from the first tree to node 4 from the second tree.

![](https://assets.leetcode.com/uploads/2024/09/24/3982-1.png)

Example 2:

Input: edges1 = [[0,1],[0,2],[0,3],[0,4]], edges2 = [[0,1],[1,2],[2,3]], k = 1

Output: [6,3,3,3,3]

Explanation:

For every `i`, connect node `i` of the first tree with any node of the second tree.

![](https://assets.leetcode.com/uploads/2024/09/24/3928-2.png)

Constraints:

* `2 <= n, m <= 1000` * `edges1.length == n - 1` * `edges2.length == m - 1` * `edges1[i].length == edges2[i].length == 2` * `edges1[i] = [ai, bi]` * `0 <= ai, bi < n` * `edges2[i] = [ui, vi]` * `0 <= ui, vi < m` * The input is generated such that `edges1` and `edges2` represent valid trees. * `0 <= k <= 1000`

Code Snippets

C++:

```
class Solution {
public:
vector<int> maxTargetNodes(vector<vector<int>>& edges1, vector<vector<int>>&
edges2, int k) {

}
};
```

Java:

```
class Solution {
public int[] maxTargetNodes(int[][] edges1, int[][] edges2, int k) {

}
}
```

Python3:

```
class Solution:
def maxTargetNodes(self, edges1: List[List[int]], edges2: List[List[int]], k:
int) -> List[int]:
```