UCF Local Contest (Final Round) — September 19, 2020

Safe Logging

filename: `logging`
Difficulty Level: `Hard`
Time Limit: `5 seconds`

Consider a tree (not necessarily a binary tree). We will use the term *neighbor* to refer to the parent and children of a node, i.e., nodes with direct connection to the node. Some of the nodes in the tree have a log. All logs have two colors: Red at the top-half and Black at the bottom-half.

Our strong friend, Lumber-Knight (LK), needs to cut all these logs in half. When a log is cut, the Red (top) part falls into a neighbor but the neighbor cannot contain a Black part. This means each node will eventually contain only one Black log or several (zero or more) Red logs. Note that the Black logs do not move from node to node. Note also that when a Red log falls into a neighbor, it does not move anymore.

Lumber-Knight, besides being strong, is also very stylish. He does not want any node with Black log to have two (or more) neighbors with Red logs. Your job is to determine all different ways the Red logs can be moved to (cut and fall into) neighbors to meet the LK's style. Since the number of ways can be quite large, output the number of ways modulo 1,000,000,007.

The Problem:

Given a tree description and locations of the logs, determine the number of ways in which Lumber-Knight can cut the logs and move the Red logs into neighbors such that no node with Black log has two (or more) neighbors with Red logs.

The Input:

The first input line contains an integer, n ($1 \le n \le 100{,}000$), representing the number of nodes in the tree (assume the tree nodes are numbered 1 to n, with 1 being the tree root). Each of the following $n - 1$ input lines contains two integers, a_i and b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$), representing a direct connection between two nodes.

Following the tree description will be an integer, m ($1 \le \mathrm{m} \le n$), representing the number of tree nodes with a log. The following input line contains m distinct integers (i.e., no duplicates), representing the tree nodes that contain a log.

The Output:

Print a single integer representing the number of ways the m logs can be cut and the Red part moved to a neighbor such that no node with a Black log has two (or more) neighbors with Red part. If it is not possible to accomplish the task, print 0 (zero).

Two cutting sequences are considered different if there is a node with different contents in the two sequences, e.g., the node is empty in one case and contains Red log(s) in the other case.

Sample Input	**Sample Output**

5
1 2
2 3
3 4
4 5
3
1 3 4
	1

6
1 2
2 3
3 4
3 5
5 6
2
2 5
	2