2022 Canadian Computing Olympiad
Day 2, Problem 1
Bi-ing Lottery Treekets

Time Limit: 1 second

Problem Description

In a parallel universe, everyone scored perfect on the CCO. As a result, Troy needs to pick the winner based on a lottery. Each contestant will choose numbers to create a ticket. A ticket is an array of size N indexed from 1 to N where each entry is a number from 0 to K.

The winning ticket is determined by dropping K balls (numbered from 1 to K) in a random sequence into a rooted binary tree. The tree has N nodes (numbered from 1 to N) and is rooted at node 1.

Each ball has a designated drop node that it will drop at. When a ball is dropped at an unoccupied node or enters an unoccupied node, one of three things happens:

1. If all of the current node's children are occupied by balls (or if a node has no children), the current ball rests at the current node. That is, it remains there and does not move again.
2. If the current node only has one unoccupied child, the current ball will move to this child.
3. If the current node has two unoccupied children, and if the current ball was just dropped, it could go either left or right. Otherwise, it will continue in the direction of its previous movement.

If all K balls cannot be dropped, a winning ticket is not determined. This happens when a ball is dropped and its drop node is occupied by another ball.

If all K balls have been dropped, the balls' resting positions determine the winning lottery ticket. The ith entry of the winning lottery ticket is the number of the ball that rests at node i or 0 if no ball rests at node i.

Troy would like to know the number of possible winning tickets (which could be zero).

Input Specification

The first line contains two space-separated integers N and K, denoting the number of nodes in the binary tree and the number of balls, respectively.

The next line contains K space-separated integers, where the ith integer denotes the designated drop node of the ball numbered i.

The last N lines each contain two space-separated integers. The ith line contains L_i and R_i denoting the ith node's left and right child, respectively, where 0 means no such child exists.

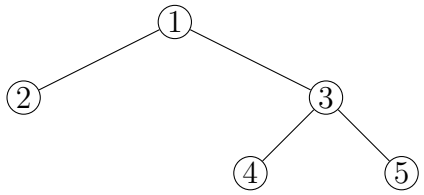
Marks Awarded	Bounds on N	Bounds on K	Additional constraints
3 marks	$1 \leq N \leq 12$	$1 \leq K \leq 6$	None
4 marks	$1 \leq N \leq 4000$	$1 \leq K \leq 4000$	All nodes do not have a left child
9 marks	$1 \leq N \leq 4000$	$1 \leq K \leq 4000$	The N drop nodes are distinct
9 marks	$1 \leq N \leq 4000$	$1 \leq K \leq 4000$	None

Output Specification
Output the remainder of the number of winning lottery tickets divided by $10^9 + 7$.

Sample Input 1
```
5 2
1 3
2 3
0 0
4 5
0 0
0 0
```

Output for Sample Input 1
```
4
```

Explanation of Output for Sample Input 1
The tree looks like this:



Consider when ball 1 is dropped first. If ball 1 goes left, then it will rest at node 2. Afterward, ball 2 is dropped and can rest at node 4 or 5. If ball 1 goes right, it will rest at node 5. Then, ball 2 will rest at node 4.

Consider when ball 2 is dropped first. Ball 2 can go left or right, resting at nodes 4 or 5, respectively. Then if ball 1 moves left after being dropped, it will rest at node 2. However, if ball 1 moves right, it will rest at either node 4 or 5, whichever place ball 2 does not occupy.

The possible winning tickets are [0, 1, 0, 2, 0], [0, 1, 0, 0, 2], [0, 0, 0, 1, 2], and [0, 0, 0, 2, 1].

Sample Input 2

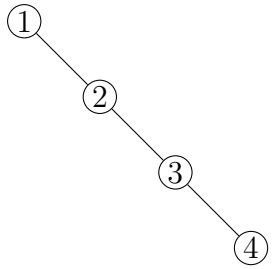
```
4 3
1 2 4
0 2
0 3
0 4
0 0
```

Output for Sample Input 2

```
2
```

Explanation of Output for Sample Input 2

The tree looks like this:



This test case satisfies the second subtask.

Ball 3 must be dropped first because if either ball 1 or ball 2 are dropped before ball 3, it would rest at node 4, which wouldn't allow ball 3 to be dropped.

Thus, we can either first drop ball 3, then ball 2 and finally ball 1 or we can first drop ball 3, then ball 1 and finally ball 2.

The possible winning tickets are [0,1,2,3] and [0,2,1,3].

Formal Definitions

A *binary tree* is a set of nodes that is either empty, or a root node with a left subtree and a right subtree both of which are binary trees. Given a node x, if its left subtree is not empty, then the root of that subtree is called the *left child* of x. Similarly, given a node x, if its right subtree is not empty, then the root of that subtree is called the *right child* of x.