First, It can be seen that after the first move the problem divides into some subproblems for each subtree that players can play in parallel, so for solving this problem we need to use grundy number.
Then consider we want to calculate grundy number of v's subtree, by using induction we can assume that we have grundy number of all subtrees of vertices like u, which u is in v's subtree.
The first and a slow approach is to fix all possible paths starting from v to its subtree and for each path calculate xor of all subtree's grundy numbers which remains after deleting the path and add this number to set S_v, then mex of S_v will be grundy number of v's subtree. this approach complexity will be $O(n^3)$ that is not fast enough.
To improve the complexity, let's use sack. For each vertex u in the subtree of v, put xor of all subtrees created if we remove path v to u in the sack of vertex v. Implement the sack as a trie with these features:

- Keep a lazy value to make it possible to xor all of the values in the sack with some number.
- A merge operation that merges the smaller trie into the greater.
- Find mex.

Now, for each vertex v merge sacks of its children (with needed changes to lazy) and find the grundy number of v.