```cpp
//
//   There is a tree with N vertices, numbered 1,2, ... N.
//   For each i (1 <= i <= N-1), the i-th edge connects Vertex x[i] and y[i].
//   Taro has decided to paint each vertex in white or black.
//   Here, it is not allowed to paint two adjacent vertices both in black.
//
//   Find the number of ways in which the vertices can be painted, modulo 10^9 + 7.
//
//      Time Complexity: O(N)
//

#include <bits/stdc++.h>
#define ll long long

using namespace std;

const int MOD = 1e9 + 7;

vector<vector<int>> adj;
vector<vector<int>> dp;

int dfs(int node, int black, int parent = -1) {
    if (dp[node][black] != -1) return dp[node][black];

    dp[node][black] = 1;
    for (int i = 0; i < (int)adj[node].size(); i++) {
        int next_node = adj[node][i];

        if (next_node == parent) continue;

        if (black) {
            dp[node][1] = ((ll)dp[node][1] * dfs(next_node, 0, node)) % MOD;
        } else {
            ll subtree_white = dfs(next_node, 0, node);
            ll subtree_black = dfs(next_node, 1, node);

            dp[node][0] = ((ll)dp[node][0] *
                ((subtree_white + subtree_black) % MOD)) % MOD;
        }
    }

    return dp[node][black];
}

int main() {
    int n;
    cin >> n;

    adj.resize(n);
    dp.resize(n, vector<int>(2, -1));

    for (int i = 0; i < n-1; i++) {
        int x, y;
        cin >> x >> y;
        x--; y--;
        adj[x].push_back(y);
        adj[y].push_back(x);
    }

    cout << ((ll)dfs(0, 0) + dfs(0, 1)) % MOD << endl;
    return 0;
}
```