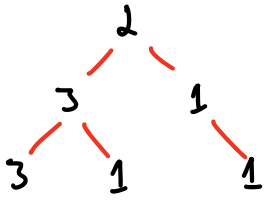
1457. Pseudo-Palindromic Paths in a Binary Tree

Given a binary tree w/ values 1-9. A path in a binary tree is palindromic if atleast one permutation of the node values is a palindrome.

Return # pseudo-palindromic paths from root → leaf nodes.

Ex.

Input:



Output: 2

Reason: 3 leaves

Paths: [2 3 3] → [2 3 2] +1

[2 3 1] → N/A +0

[2 1 1] [1 2 1] +1

Steps:

1. Produce a list of each possible path from root → leaf
2. Check if the values can produce a palindrome.
3. If yes, augment assign 1 to result.
   otherwise, do nothing
4. Final value of result is answer. result initially set to zero.

```
auto pseudoPalindromicPaths(TreeNode* root) -> int {
    decltype(auto) result{0};
    const auto paths{ produce Paths(root) };
    for (const auto& path : paths) {
        if (canBePalindrome(path)) {
            result += 1;
        }
    }
    return result;
}

// anonymous namespace should be @ top of file.
namespace {

auto producePaths(TreeNode* node) -> std::vector<std::vector<int>> {
    decltype(auto) result;
    helper(node, result);
    return result;
}

void helper(TreeNode* node, auto& result, std::optional<std::vector<int>>&
                                                    arr = std::nullopt) {
    if (!node) {
        //result.push_back(arr); // error
        return;
    }
    if (!arr) {
        arr = std::vector<int>;
    }
```

```
    arr.push_back(node->val);

    helper(node->left, result, arr);
    helper(node->right, result, arr);

    if (!node->left && !node->right) {
        result.push_back(arr); // can instead directly call `canBePalindrome`
    }

    arr.pop_back();
}

auto canBePalindrome(const auto& path) -> bool {
    // if > 1 count is odd, instantly false.
    // [1 1 2 3 3 3] 2 odds  1 2 1
    // I think otherwise true.
    std::unordered_map<int,int> m;
    for (auto val : path) {
        m[val] += 1;                                  1 1 2 2
    }                                                   1 2 2 1
    bool oddFound{};
    for (const auto& [key, value] : m) {
        if (value % 2 == 1) {
            if (oddFound) {
                return false;
            }
            oddFound = true;
        }
    }
    return true;
```

```
}

} // namespace
```

Solution worked, but slightly different from above due to Time Exceeded.

Change was to make arr unordered map, update count instead of push and pop List. Check if any counts > 1 of odds for when time to check palindrome.