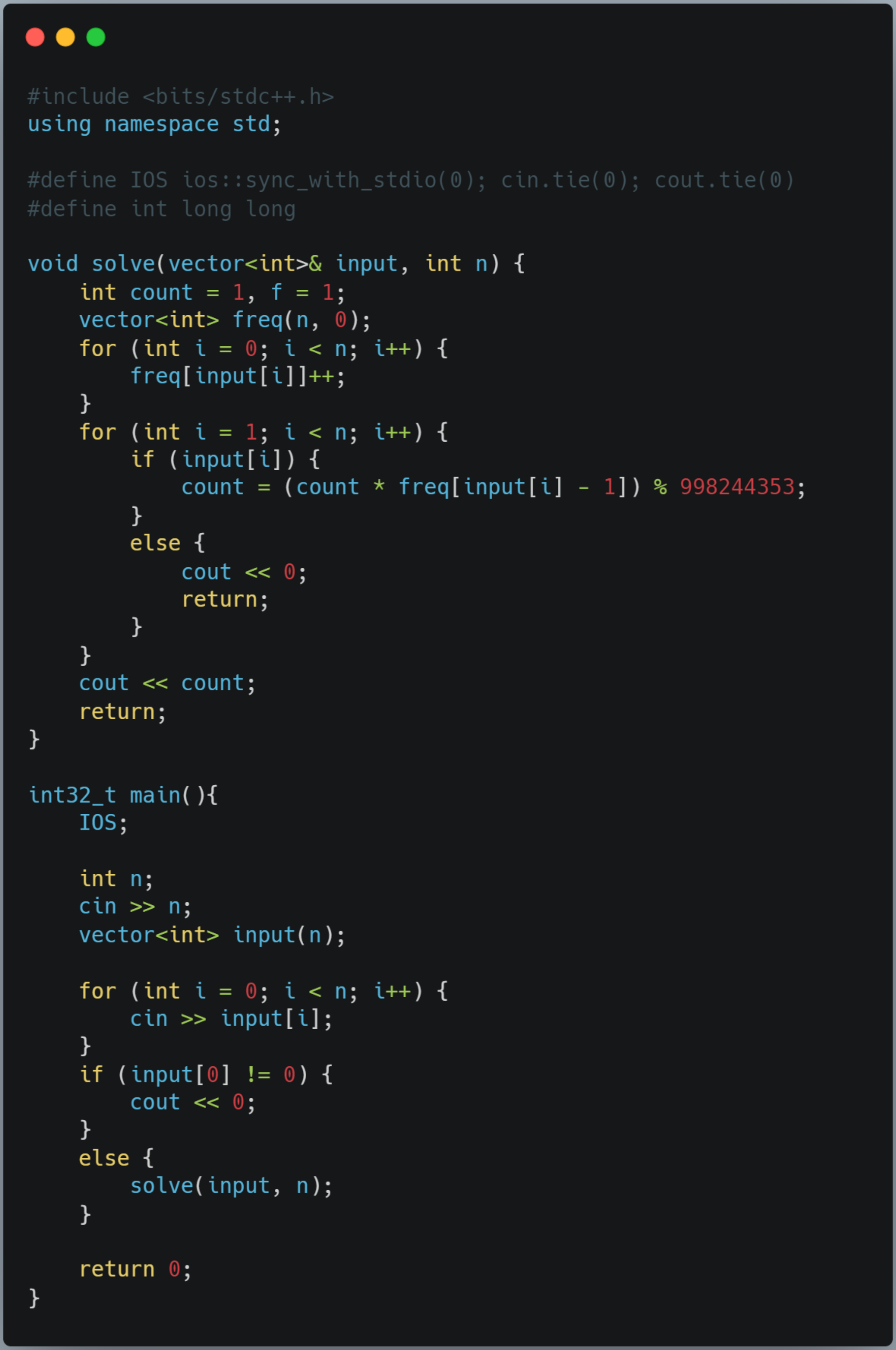
Number of special trees

Explanation: First of all, distance of a node from itself is zero, so if distance of first node from itself is not given as 0, simply our answer is 0. Now remains the other test cases where distance(1, 1) is equal to 0. Let us consider first node as root. For first node we have a single choice, so initialize the count variable as 1. Now for all the other nodes, if distance of a node is d and k number of elements have distance d – 1 from first node, then we can append this node (node with distance d) in tree in k ways, so update count by multiplying it by k. Note if d is equals to zero, then number of special trees would be 0 as no node other than the first node can have its distance from first node equals to 0.

Time Complexity- O(N)

Space Complexity- O(N)

Code:



```cpp
#include <bits/stdc++.h>
using namespace std;

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0)
#define int long long

void solve(vector<int>& input, int n) {
    int count = 1, f = 1;
    vector<int> freq(n, 0);
    for (int i = 0; i < n; i++) {
        freq[input[i]]++;
    }
    for (int i = 1; i < n; i++) {
        if (input[i]) {
            count = (count * freq[input[i] - 1]) % 998244353;
        }
        else {
            cout << 0;
            return;
        }
    }
    cout << count;
    return;
}

int32_t main(){
    IOS;

    int n;
    cin >> n;
    vector<int> input(n);

    for (int i = 0; i < n; i++) {
        cin >> input[i];
    }
    if (input[0] != 0) {
        cout << 0;
    }
    else {
        solve(input, n);
    }

    return 0;
}
```