Nested Segments

Problem

Given an array of 2n numbers, each number from 1 to n in it occurs exactly twice. We say that the segment y is nested inside the segment x if both occurrences of the number y are between the occurrences of the number x. Find for each segment i how many segments that are nested inside it.

Constraints

$$1 <= n <= 10^5$$

Example input

```
5
5 1 2 2 3 1 3 4 5 4
```

Output

```
1 0 0 0 3
```

Approach

Slight modification in 'present sir' approach.

1. Sort all the intervals in increasing order of 'r' values.
2. Start from the left and after calculating ans for each interval, mark the 'l' on the number line as present.
3. Keep updating the query's response in the answer array.

Code

```
#include "bits/stdc++.h"
using namespace std;
#define int long long
const int N = 2e5+2, MOD = 1e9+7;

int tree[4*N];

struct triplet{
```

```
    int l,r,idx;
};

int query(int node, int st, int en, int l, int r){
    if(st>r || en<l)
        return 0;

    if(l<=st && en<=r)
        return tree[node];

    int mid = (st + en)/2;
    int q1 = query(2*node, st, mid, l, r);
    int q2 = query(2*node+1, mid+1, en, l, r);

    return q1 + q2;
}

void update(int node, int st, int en, int idx, int val){
    if(st == en){
        tree[node] = val;
        return;
    }

    int mid = (st+en)/2;
    if(idx <= mid){
        update(2*node, st, mid, idx, val);
    }
    else
    {
        update(2*node+1, mid+1, en, idx, val);
    }

    tree[node] = tree[2*node] + tree[2*node+1];
}

bool compare(triplet t1, triplet t2){
    return t1.r < t2.r;
}

signed main()
```

```
{
    int n;
    cin >> n;
    triplet t1;
    t1.l = t1.r = -1;
    vector<triplet> t(n,t1);

    int x;

    for(int i=0; i<2*n; i++){
        cin >> x;

        if(t[x-1].l == -1){
            t[x-1].l = i;
        }
        else
        {
            t[x-1].r = i;
        }
        t[x-1].idx = x;
    }

    sort(t.begin(), t.end(), compare);

    vector<int> ans(n);

    for(int i=0; i<n; i++){
        ans[t[i].idx-1] = query(1,0,2*n-1, t[i].l, t[i].r);
        update(1,0,2*n-1,t[i].l, 1);
    }

    for(int i=0; i<n; i++){
        cout << ans[i] <<" ";
    }
    return 0;
}
```