Segment Tree Lazy Propagation

Source: * Shafayet'sPlanet * Shafayet'sPlanet * Source Code

```
#define mx 10000
using namespace std;
using ll = long long;

ll arr[mx];
struct info {
    ll prop, sum;
} tree[mx * 4];

void init(int node, int b, int e)
{
    if (b == e) {
        tree[node].sum = arr[b];
        return;
    }
    int Left = node * 2;
    int Right = node * 2 + 1;
    int mid = (b + e) / 2;
    init(Left, b, mid);
    init(Right, mid + 1, e);
    tree[node].sum = tree[Left].sum + tree[Right].sum;
}

void update(int node, int b, int e, int i, int j, ll x)
{
    if (i > e || j < b)
        return;
    if (b >= i && e <= j)
    {
        tree[node].sum += ((e - b + 1) * x);
        tree[node].prop += x;
        return;
    }
    int Left = node * 2;
    int Right = (node * 2) + 1;
    int mid = (b + e) / 2;
    update(Left, b, mid, i, j, x);
    update(Right, mid + 1, e, i, j, x);
    tree[node].sum = tree[Left].sum + tree[Right].sum + (e - b + 1) * tree[node].prop;
}

ll query(int node, int b, int e, int i, int j, ll carry = 0)
{
    if (i > e || j < b) return 0;
    if (b >= i and e <= j) return tree[node].sum + carry * (e - b + 1);

    int Left = node << 1;
    int Right = (node << 1) + 1;
    int mid = (b + e) >> 1;

    ll p1 = query(Left, b, mid, i, j, carry + tree[node].prop);
    ll p2 = query(Right, mid + 1, e, i, j, carry + tree[node].prop);

    return p1 + p2;
}

int main() {
//  ios_base::sync_with_stdio(false);
//  cin.tie(NULL);
    int n;
    cin >> n;
    for (int i = 1; i < n; ++i) {
        cin >> arr[i];
    }

    init(1, 1, n);
    update(1, 1, n, 1, 7, 2);
    update(1, 1, n, 1, 4, 3);
    cout << "Value for range [1, 1]: " <<  query(1, 1, n, 1, 4) << "\n\n";

    cout << "State of Segment Tree:\n";
    for (int i = 1; i < 14; ++i) {
        cout << "index: " << i << " sum: " << tree[i].sum << " prop: " << tree[i].prop << "\n";
    }


    return 0;
}
```