1807D - Odd Queries

Ferrel Destatiananda Edwardo

23 April 2024

1 Problem

Problem Description : https://codeforces.com/problemset/problem/1807/D

2 Objective

The objective of this problem is to determine whether changing all elements in specified ranges of an array to a given value will result in the sum of the entire array being odd.

3 Solution

The solution is using segment tree. 1. Initialize a segment tree with the given array. 2. When we input the array, we compute the sum of all elements in the array. 3. For every query, we first subtract the sum of elements within the query range from the total initial sum of the array and we the replacements. 4. Combining these steps, we obtain the sum of the entire array after applying the replacements, represented as (sum - query + replacements). 4. If the sum of query is odd, we output "YES". Otherwise, we output "NO".

4 Code

```
#include<bits/stdc++.h>

using namespace std;

struct segmenttree {
        int n;
        vector<int> st;

        void init(int _n) {
                this->n = _n;
                st.resize(4 * n, 0);
        }

        void build(int start, int ending, int node, vector<int> &v) {
                // leaf node base case
                if (start == ending) {
                        st[node] = v[start];
                        return;
                }

                int mid = (start + ending) / 2;

                // left subtree is (start,mid)
                build(start, mid, 2 * node + 1, v);

                // right subtree is (mid+1,ending)
                build(mid + 1, ending, 2 * node + 2, v);

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

        int query(int start, int ending, int l, int r, int node) {
                // non overlapping case
                if (start > r || ending < l) {
                        return 0;
                }

                // complete overlap
                if (start >= l && ending <= r) {
                        return st[node];
                }

                // partial case
```

```
                int mid = (start + ending) / 2;

                int q1 = query(start, mid, l, r, 2 * node + 1);
                int q2 = query(mid + 1, ending, l, r, 2 * node + 2);

                return q1 + q2;
        }


        void build(vector<int> &v) {
                build(0, n - 1, 0, v);
        }

        int query(int l, int r) {
                return query(0, n - 1, l, r, 0);
        }

};

int main() {
    int t;
    cin >> t;
    while (t--) {
        int n, q;
        cin >> n >> q;
        vector<int> v(n);
        int sum = 0;
        for (int i = 0; i < n; i++) {
            cin >> v[i];
            sum += v[i];
        }

        segmenttree tree;
        tree.init(n);
        tree.build(v);

        for (int i = 0; i < q; i++) {
            int l, r, k;
            cin >> l >> r >> k;
            if ((sum - tree.query(l - 1, r - 1) + k * (r - l + 1)) % 2 != 0)
                cout << "YES" << endl;
            } else {
                cout << "NO" << endl;
            }
        }
    }
```

```
    return 0;
}
```