Competitive Programming Note

Moon Jam

March 4, 2024

Contents

1 使用場景

- 區間修改、區間查詢：線段樹＋懶人標記
- 區間修改、單點查詢：BIT＋差分、線段樹＋懶人標記
- 單點修改、區間查詢：線段樹、BIT
- 區間修改、沒有查詢：差分
- 區間查詢、沒有修改：前綴和、稀疏表

2 Segment Tree

```
#define lc id * 2 + 1
#define rc id * 2 + 2
#define M ((L + R) / 2)

struct Nodes {
    Info info;
    Tag tag;
} seg[N * 4];

void push(int L, int R, int id) {
    if (R > L + 1) {
        seg[lc].tag ... seg[id].tag;
        seg[rc].tag ... seg[id].tag;
    }
    seg[id].info = ... seg[id].tag;  // Update the info from tag
    seg[id].tag = NULL;              // clear seg[id].tag
}

void pull(int L, int R, int id) {
    push(L, M, lc), push(M, R, rc);
    // Update the info of id from lc and rc
    seg[id].info = seg[lc].info ... seg[rc].info;
}

void modify(int l, int r, Tag tag, int L = 0, int R = n, int id = 0) {
    push(id);
    if (l >= L && r <= R) {
        seg[id].tag = tag;
    } else {
        if (l >= M)
            modify(l, r, tag, M, R, rc);
        else if (r =< M)
            modify(l, r, tag, L, M, lc);
        else
            modify(l, M, tag, L, M, lc), modify(M, r, tag, M, R, rc);
        pull(id);
    }
}

Info query(int l, int r, int L = 0, int R = n, int id = 0) {
    push(id);
    if (l == L && r == R) return id.info;
    if (l >= M) return query(l, r, M, R, rc);
    if (r <= M) return query(l, r, lc);
    // calculate answer from lc and rc
    return query(l, M, L, M, lc) ... query(M, r, M, R, rc);
}
```

3 Binary Indexed Tree

4 Sparse Table

5 莫隊

- 可以離線處理(沒有修改)
- 區間眾數、區間最大連續和、區間某個數字的數量

Range Pairing Query

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

int n, q, s;

struct qu {
    int l, r, id;
    bool operator<(const qu &B) const {
        if (l / s != B.l / s) return l < B.l;
        return (l / s & 1) ? r < B.r : r > B.r;
    }
};

signed main() {
    cin >> n;
    vector<int> arr(n + 1);
    for(int i = 1; i<=n; i++) cin >> arr[i];
    s = sqrt(n);

    cin >> q;
    vector<qu> qus(q);
    for(int i = 0; i<q; i++) {
        int a, b;
        cin >> a >> b;
        qus[i] = {a, b, i};
    }
    sort(qus.begin(), qus.end());

    int l = 1, r = 1, p = 0;
    set<pair<int, int>> ans;
    vector<int> cur(n + 1);
    cur[arr[1]] = 1;
    for (qu i : qus) {
        while (i.r > r) {
            r++;
            if (cur[arr[r]] & 1) p++;
            cur[arr[r]]++;
        }
        while (i.l > l) {
            cur[arr[l]]--;
            if (cur[arr[l]] & 1) p--;
            l++;
        }
        while (i.r < r) {
            cur[arr[r]]--;
            if (cur[arr[r]] & 1) p--;
            r--;
        }
        while (i.l < l) {
            l--;
            if (cur[arr[l]] & 1) p++;
```

```
            cur[arr[l]]++;
        }
        ans.insert({i.id, p});
    }
    for (auto i : ans) cout << i.second << '\n';
    return 0;
}
```

6 DSU

7 PBDS & ROPE

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

template <class T>
using Tree =
    tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;
template <class T>
using multiTree =
    tree<T, null_type, less_equal<T>, rb_tree_tag, tree_order_statistics_node_update>;

int main() {
    Tree<int> tr;
    tr.insert(1), tr.insert(8), tr.insert(3);
    // an iterator of the (k+1)th smallest element
    cout << *tr.find_by_order(1) << '\n';
    // the number of elements are less than ktyl,
    cout << tr.order_of_key(6) << '\n';
    tr.erase(8);
    return 0;
}
```

```cpp
#include <bits/extc++.h>
using namespace __gnu_cxx;
using namespace std;
int main() {
    rope<int> r, a, b;
    b += 10;          // {10}
    r += 2;           // {2}
    r += 3;           // {2, 3}
    a = r + r + b;    // {2, 3, 2, 3, 10}
    for (int i : a) cout << i << '-';
    cout << '\n';
    b += a.substr(2, 3);  // get 3 elements from a_2
    for (int i : b) cout << i << '-';
    cout << '\n';
    r.pop_back();         // {3}
    r.pop_front();        // empty
    r.push_back(1);       // {1}
    r.push_front(10);     // {10, 1}
    r.replace(1, 100);    // replace r_1 to 100
    r.erase(0, 1);        // erase 1 element from 0
    r.insert(0, 7);       // insert 7 at r_0
    for (int i : r) cout << i << '-';
}
```

8 拓樸排序

9 Floyd

10 Dijkstra

11 Bellman-Ford

12 Prime

13 Kruskal

14 Tarjan

15 凸包