莫队习题

38578 「SPOJ3267」DQUERY - D-query

Solution

只需要一个 `cnt` 数组记录元素个数即可，数量可以很简单借此统计出来

注意当**值域**较大时可以**离散化**

Code

```
void solve() {
    int n; cin >> n;
    int sq = sqrt(n) + 1; cmin(sq, n);
    vector a(n, 0); for (auto& v: a) cin >> v;

    vector pool = a;
    sort(all(pool)); pool.erase(unique(all(pool)), pool.end());
    for (auto& v: a) v = lower_bound(all(pool), v) - pool.begin();

    struct Node { int l, r, id; };

    int q; cin >> q;
    vector Q(q, Node());
    for (int i = 0; i < q; i++) {
        cin >> Q[i].l >> Q[i].r;
        Q[i].l--; Q[i].r--;
        Q[i].id = i;
    }

    sort(all(Q), [&](const Node& a, const Node& b){
        if(a.l / sq != b.l / sq) return a.l < b.l;
        if((a.l / sq) & 1)return a.r < b.r;
        else return b.r < a.r;
    });

    int cur = 0;
    vector ans(q, 0), cnt(n, 0);

    auto add = [&](int p) {
        cur += ((cnt[a[p]]++) == 0);
    };

    auto del = [&](int p) {
        cur -= ((--cnt[a[p]]) == 0);
    };

    int L = 1, R = 0;
    for (auto [l, r, id]: Q) {
        while(L < l) del(L++);
        while(L > l) add(--L);
        while(R < r) add(++R);
        while(R > r) del(R--);
        ans[id] = cur;
```

```
    }

    for (auto v: ans) cout << v << '\n';
}
```

3411 小B的询问

Solution

我们仍然统计个数，注意到 $(x+1)^2 = x^2 + 2x + 1$，因此我们可以在 `add` 操作时，用加法加上对应的值，反之亦然（常数比乘法好一些）

Code

```
void solve() {
    int n, q, k; cin >> n >> q >> k;
    int sq = sqrt(n) + 1; cmin(sq, n);
    vector a(n, 0); for (auto& v: a) cin >> v;

    vector pool = a;
    sort(all(pool)); pool.erase(unique(all(pool)), pool.end());
    for (auto& v: a) v = lower_bound(all(pool), v) - pool.begin();

    struct Node { int l, r, id; };

    vector Q(q, Node());
    for (int i = 0; i < q; i++) {
        cin >> Q[i].l >> Q[i].r;
        Q[i].l--; Q[i].r--;
        Q[i].id = i;
    }

    sort(all(Q), [&](const Node& a, const Node& b){
        if(a.l / sq != b.l / sq) return a.l < b.l;
        if((a.l / sq) & 1)return a.r < b.r;
        else return b.r < a.r;
    });

    ll cur = 0;
    vector ans(q, 0ll), cnt(n, 0ll);

    auto add = [&](int p) {
        cnt[a[p]]++;
        cur += cnt[a[p]] * 2 - 1;
    };

    auto del = [&](int p) {
        cnt[a[p]]--;
        cur -= cnt[a[p]] * 2 + 1;
    };

    int L = 1, R = 0;
    for (auto [l, r, id]: Q) {
        while(L < l) del(L++);
```

```
        while(L > l) add(--L);
        while(R < r) add(++R);
        while(R > r) del(R--);
        ans[id] = cur;
    }

    for (auto v: ans) cout << v << '\n';
}
```

6089 小Z的袜子

Solution

总共有 n 只袜子的情况下，我们有 $n(n-1)$ 种选法（考虑先后顺序）

在同种颜色袜子有 k_i 只的情况下，我们有 $k_i(k_i-1)$ 种选法

那么答案就是 $\frac{\sum(k_i(k_i-1))}{n(n-1)}$

显然，只需要维护每种颜色的袜子的只数就可以了

对于计算难度稍大的题，我们可以在 `add` 或 `del` 操作之前把原答案删去，等操作完成后再统计一次答案

Code

```
void solve() {
    int n, q; cin >> n >> q;
    int sq = sqrt(n) + 1; cmin(sq, n);
    vector a(n, 0); for (auto& v: a) cin >> v;

    vector pool = a;
    sort(all(pool)); pool.erase(unique(all(pool)), pool.end());
    for (auto& v: a) v = lower_bound(all(pool), v) - pool.begin();

    struct Node { int l, r, id; };

    vector Q(q, Node());
    for (int i = 0; i < q; i++) {
        cin >> Q[i].l >> Q[i].r;
        Q[i].l--; Q[i].r--;
        Q[i].id = i;
    }

    sort(all(Q), [&](const Node& a, const Node& b){
        if(a.l / sq != b.l / sq) return a.l < b.l;
        if((a.l / sq) & 1)return a.r < b.r;
        else return b.r < a.r;
    });

    ll cur = 0;
    vector cnt(n, 0ll);
    vector ans(q, pair(0ll, 0ll));
```

```
    auto add = [&](int p) {
        cur -= cnt[a[p]] * (cnt[a[p]] - 1);
        cnt[a[p]]++;
        cur += cnt[a[p]] * (cnt[a[p]] - 1);
    };

    auto del = [&](int p) {
        cur -= cnt[a[p]] * (cnt[a[p]] - 1);
        cnt[a[p]]--;
        cur += cnt[a[p]] * (cnt[a[p]] - 1);
    };

    int L = 1, R = 0;
    for (auto [l, r, id]: Q) {
        while(L < l) del(L++);
        while(L > l) add(--L);
        while(R < r) add(++R);
        while(R > r) del(R--);
        ans[id] = {cur, 1ll * (r - l + 1) * (r - l)};
    }

    for (auto [f, s]: ans) {
        if (!f) cout << "0/1\n";
        else {
            auto g = gcd(f, s);
            cout << f / g << '/' << s / g << '\n';
        }
    }
}
```

47692 「HDU6959」ZOTO

Solution

如果不考虑 y 这一维的话，实际上就是个很简单的**去重**问题

但是加上 y 这一维的限制的话，我们就需要对 cnt 进行一定的处理

将 `add` 和 `del` 问题转化为对 cnt 数组的单点修改，将 `qry` 问题转化为对 cnt$ 数组的区间查询

显然可以用**分块**维护 cnt 数组

`add` `del` 操作复杂度仍为 $O(1)$，查询复杂度为 $O(\sqrt{n})$，总时间复杂度为 $O(n\sqrt{n})$

Code

```
void solve() {
    int n, q; cin >> n >> q;;
    int sq = sqrt(maxm) + 1; cmin(sq, maxm);
    vector a(n, 0); for (auto& v: a) cin >> v;
    vector bk(maxm / sq + 2, 0);

    struct Node { int l1, r1, l2, r2, id; };

    vector Q(q, Node());
```

```
    for (int i = 0; i < q; i++) {
        cin >> Q[i].l1 >> Q[i].l2 >> Q[i].r1 >> Q[i].r2;
        Q[i].l1--; Q[i].r1--;
        Q[i].id = i;
    }

    sort(all(Q), [&](const Node& a, const Node& b){
        if(a.l1 / sq != b.l1 / sq) return a.l1 < b.l1;
        if((a.l1 / sq) & 1)return a.r1 < b.r1;
        else return b.r1 < a.r1;
    });

    int cur = 0;
    vector ans(q, 0), cnt(n, 0);

    auto add = [&](int p) {
        if (++cnt[a[p]] == 1) bk[a[p] / sq]++;
        cur++;
    };

    auto del = [&](int p) {
        if (--cnt[a[p]] == 0) bk[a[p] / sq]--;
        cur++;
    };

    auto qry = [&](int l, int r) {
        int ret = 0;
        while (l <= r && l % sq) ret += (cnt[l++] > 0);
        while (r >= l && r % sq != sq - 1) ret += (cnt[r--] > 0);
        l = l / sq, r = (r + 1) / sq;
        for (int i = l; i < r; i++) ret += bk[i];
        return ret;
    };

    int L = 1, R = 0;
    for (auto [l1, r1, l2, r2, id]: Q) {
        while(L < l1) del(L++);
        while(L > l1) add(--L);
        while(R < r1) add(++R);
        while(R > r1) del(R--);
        ans[id] = qry(l2, r2);
    }

    for (auto v: ans) cout << v << '\n';
}
```

35137 「CF617E」XOR and Favorite Number

Solution

首先，观察到异或操作具有**前缀和**特性，即，记前 i 个数的异或和为 $sum[i]$ ，则有：

$$a_x \oplus a_{x+1} \cdots a_y = sum[x-1] \oplus sum[y]$$

因此，我们首先将整个数列进行异或前缀和操作（特殊的，我们记 $a[0] = 0$）

在尾部加一个数时，我们查看有多少地方 $sum[i] = sum[R] \oplus k$ ，这个很容易用计数数组实现

在头部删一个数时也是类似

注意特殊处理 $sum[R] \oplus k = sum[R]$ 的情况即可

Code

```
void solve() {
    int n, q, k; cin >> n >> q >> k;
    int sq = sqrt(n) + 1; cmin(sq, n);
    vector a(n + 1, 0);
    for (int i = 1; i <= n; i++) {
        cin >> a[i]; a[i] ^= a[i - 1];
    }

    struct Node { int l, r, id; };

    vector Q(q, Node());
    for (int i = 0; i < q; i++) {
        cin >> Q[i].l >> Q[i].r;
        Q[i].l--;
        Q[i].id = i;
    }

    sort(all(Q), [&](const Node& a, const Node& b){
        if(a.l / sq != b.l / sq) return a.l < b.l;
        if((a.l / sq) & 1)return a.r < b.r;
        else return b.r < a.r;
    });

    ll cur = 0;
    vector cnt(maxm, 0ll), ans(q, 0ll);

    auto add = [&](int p) {
        cnt[a[p]]++;
        cur += cnt[a[p] ^ k];
        if ((a[p] ^ k) == a[p]) cur--;
    };

    auto del = [&](int p) {
        cnt[a[p]]--;
        cur -= cnt[a[p] ^ k];
    };

    int L = 1, R = 0;
    for (auto [l, r, id]: Q) {
```

```
        while(L < l) del(L++);
        while(L > l) add(--L);
        while(R < r) add(++R);
        while(R > r) del(R--);
        ans[id] = cur;
    }

    for (auto v: ans) cout << v << '\n';
}
```

9780 「CF221D」 Little Elephant and Array

Solution

我们仍然可以用 `cnt` 数组存储数字的出现次数，并在 `add` 和 `del` 操作时对答案进行相应的修改即可

唯一需要注意的地方是，我们的 `cnt` 数组并不需要开到 $O(\text{值域})$，也不需要离散化，因为显然，大于 n 的数不会对答案产生任何贡献

Code

```
void solve() {
    int n, q; cin >> n >> q;
    int sq = sqrt(n) + 1; cmin(sq, n);
    vector a(n, 0); for (auto& v: a) cin >> v;

    struct Node { int l, r, id; };

    vector Q(q, Node());
    for (int i = 0; i < q; i++) {
        cin >> Q[i].l >> Q[i].r;
        Q[i].l--; Q[i].r--;
        Q[i].id = i;
    }

    sort(all(Q), [&](const Node& a, const Node& b){
        if(a.l / sq != b.l / sq) return a.l < b.l;
        if((a.l / sq) & 1)return a.r < b.r;
        else return b.r < a.r;
    });

    ll cur = 0;
    vector cnt(n + 1, 0ll), ans(q, 0ll);

    auto add = [&](int p) {
        if (a[p] > n) return;
        if (a[p] == cnt[a[p]]) cur--;
        cnt[a[p]]++;
        if (a[p] == cnt[a[p]]) cur++;
    };

    auto del = [&](int p) {
        if (a[p] > n) return;
        if (a[p] == cnt[a[p]]) cur--;
```

```
        cnt[a[p]]--;
        if (a[p] == cnt[a[p]]) cur++;
    };

    int L = 1, R = 0;
    for (auto [l, r, id]: Q) {
        while(L < l) del(L++);
        while(L > l) add(--L);
        while(R < r) add(++R);
        while(R > r) del(R--);
        ans[id] = cur;
    }

    for (auto v: ans) cout << v << '\n';
}
```