Nominating Group Leaders

locked

by bertho_coder

Editorial by bertho_coder

First, we sort the group nominee queries according to MO's Algorithm and process the queries one by one. We maintain an array, $freq$, where each $freq[i]$ represents the number of times the value i appears in currently covered range. When adding or removing any number, we must update the $freq$ array in $O(1)$ time. Now while determining the nominee for a group, we can loop over the $freq$ array to find the smallest integer that appears exactly x times. However, this solution takes $O(q \cdot maxvalue)$ time in the worst case, where $maxvalue$ is the maximum value that can appear in the given array, so this approach will not satisfy the time constraints for all test cases.

Now, let's improve this solution by decomposing the $freq$ array into $\lceil \sqrt{maxvalue} \rceil$ buckets, each having at most $\sqrt{maxvalue}$ numbers and, for every bucket, let's maintain an array named $freq\_of\_freq$, where $freq\_of\_freq[i]$ represents the number of times the number i appeared in the corresponding bucket. When adding or removing numbers, we must to update this array as well. Now when we want to determine the nominee for a group, we simply loop over the buckets from left to right and check whether x appeared in a bucket or not. Once we find the bucket where x appears, we can loop over the bucket's members to find the smallest ID that appears exactly x times.

Time complexity: $\mathcal{O}((g+n)\sqrt{n})$.

Statistics

Difficulty: Expert
Time Complexity: $\mathcal{O}((g+n) \cdot sqrt(n))$
Required Knowledge: Sqrt Decomposition, MO's Algorithm
Publish Date: Dec 20 2016

Set by bertho_coder

Problem Setter's code :

```
#include<bits/stdc++.h>
using namespace std;
#define D(x)        cout << #x " = " << (x) << endl
#define MAX         100000

const int bucksz = 320;
set<int> candidates[MAX + 5];

struct query{
    int l, r, x, id;

    query(){}
    query(int _l, int _r, int _x, int _id){
        l = _l, r = _r, x = _x, id = _id;
    }
};

bool operator < (const query &u, const query &v){
    if(u.l / bucksz == v.l / bucksz) return u.r < v.r;
    return u.l < v.l;
}

struct solver{
    int n, q;
    vector<int> numbers, L, R, X;

    solver(){}
    void takeInput(){
```

```
    scanf("%d", &n);
    numbers.resize(n + 5);
    for(int i = 1; i <= n; i++)
    {
                scanf("%d", &numbers[i]);
                numbers[i]++;
        }

    scanf("%d", &q);
    L.resize(q + 5);
    R.resize(q + 5);
    X.resize(q + 5);
    for(int i = 1; i <= q; i++)
    {
                scanf("%d %d %d", &L[i], &R[i], &X[i]);
                L[i]++;
                R[i]++;
        }
}

vector<int> brute(){
    vector<int> frequency, answers;
    frequency.resize(MAX+5);
    answers.resize(q + 5);

    for(int i = 1; i <= q; i++)
    {
        for(int k = L[i]; k <= R[i]; k++)
            frequency[numbers[k]]++;
        int pos = 1;
        while(pos <= MAX && frequency[pos] != X[i]) pos++;

        answers[i] = (pos <= MAX) ? pos : -1;

        for(int k = L[i]; k <= R[i]; k++)
            frequency[numbers[k]] = 0;
    }

    return answers;
}

vector<int> O_NrtN(){
    vector<int> answers;
    answers.resize(q + 5, -1);

    vector<query> Q;
    Q.resize(q + 5);
    for(int i = 1; i <= q; i++)
        Q[i] = query(L[i], R[i], X[i], i);
    sort(Q.begin() + 1, Q.begin() + q + 1);

    int ttl_buk = MAX / bucksz + 1;
    vector<int> frequency, bucket_info[ ttl_buk + 5];
    frequency.resize(MAX + 5);
    for(int i = 1; i <= ttl_buk; i++)
        bucket_info[i].resize(MAX+5);

    int l = 0, r = 0;

    for(int i = 1; i <= q; i++)
    {
        while(l > Q[i].l){
            l--;
            int prv = frequency[numbers[l]];
            int buck_no = numbers[l] / bucksz + 1;
            bucket_info[buck_no][prv]--;
            bucket_info[buck_no][prv + 1]++;
            frequency[numbers[l]]++;
        }

        while(r < Q[i].r){
            r++;
            if(r){
                int prv = frequency[numbers[r]];
                int buck_no = numbers[r] / bucksz + 1;
                bucket_info[buck_no][prv]--;
                bucket_info[buck_no][prv+1]++;
                frequency[numbers[r]]++;
            }
        }
```

```
            while(l < Q[i].l){
                if(l){
                    int prv = frequency[numbers[l]];
                    int buck_no = numbers[l] / bucksz + 1;
                    bucket_info[buck_no][prv]--;
                    bucket_info[buck_no][prv-1]++;
                    frequency[numbers[l]]--;
                }
                l++;
            }

            while(r > Q[i].r){
                int prv = frequency[numbers[r]];
                int buck_no = numbers[r] / bucksz + 1;
                bucket_info[buck_no][prv]--;
                bucket_info[buck_no][prv-1]++;
                frequency[numbers[r]]--;
                r--;
            }

            for(int b = 1; b <= ttl_buk; b++)
                if(bucket_info[b][Q[i].x]){
                    int st = max(1, (b - 1) * bucksz);
                    while(frequency[st] != Q[i].x) st++;
                    answers[Q[i].id] = st;
                    break;
                }
        }

        return answers;
    }

    vector<int> O_NrtNLgN(){
        vector<int> answers;
        answers.resize(q + 5, -1);

        vector<query> Q;
        Q.resize(q + 5);
        for(int i = 1; i <= q; i++)
            Q[i] = query(L[i], R[i], X[i], i);
        sort(Q.begin() + 1, Q.begin() + q + 1);

        vector<int> frequency;
        frequency.resize(MAX + 5);
        for(int i = 1; i <= MAX; i++)
            candidates[i].clear();


        int l = 0, r = 0;

        for(int i = 1; i <= q; i++)
        {
            while(l > Q[i].l){
                l--;
                int prv = frequency[numbers[l]];
                if(prv) candidates[prv].erase(numbers[l]);
                candidates[prv + 1].insert(numbers[l]);
                frequency[numbers[l]]++;
            }

            while(r < Q[i].r){
                r++;
                if(r){
                    int prv = frequency[numbers[r]];
                    if(prv) candidates[prv].erase(numbers[r]);
                    candidates[prv + 1].insert(numbers[r]);
                    frequency[numbers[r]]++;
                }
            }

            while(l < Q[i].l){
                if(l){
                    int prv = frequency[numbers[l]];
                    if(prv) candidates[prv].erase(numbers[l]);
                    if(prv - 1) candidates[prv-1].insert(numbers[l]);
                    frequency[numbers[l]]--;
                }
                l++;
            }

            while(r > Q[i].r){
```

```
            int prv = frequency[numbers[r]];
            candidates[prv].erase(numbers[r]);
            if(prv - 1) candidates[prv - 1].insert(numbers[r]);
            frequency[numbers[r]]--;
            r--;
        }

        if(candidates[Q[i].x].empty()) answers[Q[i].id] = -1;
        else answers[Q[i].id] = *(candidates[Q[i].x].begin());
    }

    return answers;
}

vector<int> O_shouldNotPass(){
    vector<int> answers;
    answers.resize(q + 5, -1);

    vector<query> Q;
    Q.resize(q + 5);
    for(int i = 1; i <= q; i++)
        Q[i] = query(L[i], R[i], X[i], i);
    sort(Q.begin() + 1, Q.begin() + q + 1);

    vector<int> frequency;
    frequency.resize(MAX + 5);
    int mv = *max_element(numbers.begin() + 1, numbers.begin() + n + 1);

    int l = 0, r = 0;
    for(int i = 1; i <= q; i++)
    {
        while(l > Q[i].l){
            l--;
            int prv = frequency[numbers[l]];
            frequency[numbers[l]]++;
        }

        while(r < Q[i].r){
            r++;
            if(r){
                int prv = frequency[numbers[r]];
                frequency[numbers[r]]++;
            }
        }

        while(l < Q[i].l){
            if(l){
                int prv = frequency[numbers[l]];
                frequency[numbers[l]]--;
            }
            l++;
        }

        while(r > Q[i].r){
            int prv = frequency[numbers[r]];
            frequency[numbers[r]]--;
            r--;
        }

        int v = 1;
        while(v <= mv && frequency[v] != Q[i].x) v++;
        if(v > mv) answers[Q[i].id] = -1;
        else answers[Q[i].id] = v;
    }

    return answers;
}

void checker_O_NrtN(){
    vector<int> u = O_NrtN();
    vector<int> v = brute();
    int ret = 0;
    for(int i = 1; i <= q; i++)
        ret += (u[i] != v[i]);

    printf("Missmatches: %d\n", ret);
}

void checker_O_NrtNLgN(){
    vector<int> u = O_NrtN();
    vector<int> v = O_NrtNLgN();
```

```
        int ret = 0;
        for(int i = 1; i <= q; i++)
            ret += (u[i] != v[i]);

        printf("Missmatches: %d\n", ret);
    }

    void checker_O_shouldNotPass(){
        vector<int> u = O_NrtN();
        vector<int> v = O_shouldNotPass();
        int ret = 0;
        for(int i = 1; i <= q; i++)
            ret += (u[i] != v[i]);

        printf("Missmatches: %d\n", ret);
    }

    void print(vector<int> answers){
        for(int i = 1; i <= q; i++)
            if(answers[i] == -1) printf("%d\n", answers[i]);
                        else printf("%d\n", answers[i] - 1);
    }
}S;

int main()
{
    //freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);

    int t;
    scanf("%d", &t);
    while(t--)
    {
        S.takeInput();
        S.print(S.O_NrtN());
    }

    return 0;
}
```

Tested by Wild_Hamster

Problem Tester's code :

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <vector>
#include <cstring>
#include <deque>
#include <stack>
#include <stdio.h>
#include <map>
#include <set>
#include <time.h>
#include <string>
#include <fstream>
#include <queue>
#include <bitset>
#include <cstdlib>
#include <assert.h>
#include <list>
#include <unordered_map>
#define X first
#define Y second
#define mp make_pair
#define pb push_back
#define pdd pair<double,double>
#define pii pair<ll,ll>
#define PI 3.14159265358979323846
#define MOD 1000000007
#define MOD2 1000000009
#define INF ((ll)1e+18)
#define x1 fldgjdflgjhrthrl
#define x2 fldgjdflgrtyrtyjl
#define y1 fldggfhfghjdflgjl
#define y2 ffgfldgjdflgjl
```

```
#define SQ 317
#define MAG 33554431
#define RED 0
#define BLUE 1
#define ALP 26
typedef int ll;
typedef long double ld;
using namespace std;
ll i,j,k,l,m,r,x,y,K,tot,sz,cur,sum,n,c, maxlvl,q,z,N;
ll a[100500], ans[100500], freq[100500];
ll Dec[405][100500];
vector<ll> g[100500];
pair<pii,pii> queries[100500];
bool cmp(pair<pii,pii> x,pair<pii,pii> y)
{
    if (x.X.X/SQ > y.X.X/SQ)
        return false;
    if (x.X.X/SQ < y.X.X/SQ)
        return true;
    return (x.X.Y < y.X.Y);
}
int main() {
        //freopen("input.txt","r",stdin);
        //freopen("input.txt","w",stdout);
    ll tests;
    ll sum_n = 0, sum_q = 0;
    cin >> tests;
    assert(tests >= 1 && tests <= 5);
    while (tests--)
    {
        cin >> n;
        assert(n >= 1 && n <= 100000);
        sum_n += n;
        for (i = 0; i <= n; i++)
        {
            freq[i] = 0;
        }
        for (i = 1; i <= n; i++)
        {
            scanf("%d",&a[i]);
            a[i]++;
            assert(a[i] >= 1 && a[i] <= n);
        }
        for (i = 0; i <= n/SQ+1; i++)
            for (j = 0; j <= n; j++)
                Dec[i][j] = 0;
        cin >> q;
        assert(q >= 1 && q <= 100000);
        sum_q += q;
        for (i = 0; i < q; i++)
        {
            scanf("%d %d %d",&x,&y,&z);
            x++;y++;
            assert(1 <= x && x <= y && y <= n && 1 <= z && z  <= n);
            queries[i] = mp(mp(x,y), mp(z,i));
        }
        sort(queries, queries+q, cmp);
        ll L = queries[0].X.X, R = queries[0].X.Y;
        z = queries[0].Y.X;
        ll num = queries[0].Y.Y;
        for (j = L; j <= R; j++)
        {
            Dec[a[j]/SQ][freq[a[j]]]--;
            freq[a[j]]++;
            Dec[a[j]/SQ][freq[a[j]]]++;
        }
        ans[num] = MOD;
        for (int ii = 0; ii <= n/SQ+1; ii++)
            if (Dec[ii][z] > 0)
            {
                for (j = ii*SQ; j < ii*SQ+SQ; j++)
                    if (freq[j] == z)
                    {
                        ans[num] = j;
                        break;
                    }
                break;
            }
        for (i = 1; i < q; i++)
        {
            ll l = queries[i].X.X, r = queries[i].X.Y;
```

```
            while (L < l)
            {
                Dec[a[L]/SQ][freq[a[L]]]--;
                freq[a[L]]--;
                Dec[a[L]/SQ][freq[a[L]]]++;
                L++;
            }
            while (R > r)
            {
                Dec[a[R]/SQ][freq[a[R]]]--;
                freq[a[R]]--;
                Dec[a[R]/SQ][freq[a[R]]]++;
                R--;
            }
            while (L > l)
            {
                L--;
                Dec[a[L]/SQ][freq[a[L]]]--;
                freq[a[L]]++;
                Dec[a[L]/SQ][freq[a[L]]]++;
            }
            while (R < r)
            {
                R++;
                Dec[a[R]/SQ][freq[a[R]]]--;
                freq[a[R]]++;
                Dec[a[R]/SQ][freq[a[R]]]++;
            }
            z = queries[i].Y.X;
            ll num = queries[i].Y.Y;
            ans[num] = MOD;
            for (int ii = 0; ii <= n/SQ+1; ii++)
                if (Dec[ii][z] > 0)
                {
                    for (j = ii*SQ; j < ii*SQ+SQ; j++)
                        if (freq[j] == z)
                        {
                            ans[num] = j;
                            break;
                        }
                    break;
                }
        }
        for (i = 0; i < q; i++)
            printf("%d\n",(ans[i]==MOD?-1:ans[i]-1));
    }
    assert(sum_n >= 1 && sum_n <= 300000 && sum_q >= 1 && sum_q <= 300000);
    return 0;
}
```