带修莫队

- Write by ***Withinlover***

本文建立在已经掌握并熟知普通莫队的基础上，如果你对普通莫队的使用仍有疑惑，请前往阅读普通莫队。

算法介绍

普通莫队的一大局限在于不支持修改操作。

我们可以通过增加时间轴的方式让莫队强行支持修改。

具体的，就是由 (l, r) 变为 (l, r, t)

若已知 (l, r, t)我们考虑如下 6 种情况

- $(l - 1, r, t)$
- $(l + 1, r, t)$
- $(l, r - 1, t)$
- $(l, r + 1, t)$
- $(l, r, t - 1)$
- $(l, r, t + 1)$

若这些状态均可以由 (l, r, t) 在 $O(1)$ 或其他极短的时间内完成。那么便可以考虑带修莫队。假设 n 和 m 同阶，其复杂度为 $O\left(n^{\frac{5}{3}}\right)$

写法对比

普通莫队

```
for(int i = 1;i <= m; ++i){
    while(r < Q[i].r) ans = ans + Calc(++r, 1);
    while(r > Q[i].r) ans = ans + Calc(r--, -1);
    while(l < Q[i].l) ans = ans + Calc(l++, -1);
    while(l > Q[i].l) ans = ans + Calc(--l, 1);
    Ans[Q[i].pos] = ans;
}
```

带修莫队

```
for(int i = 1;i <= m; ++i) {
    while(r < Q[i].r) Add(a[++r]);
    while(r > Q[i].r) Del(a[r--]);
    while(l < Q[i].r) Del(a[l++]);
    while(l > Q[i].l) Add(a[--l]);
    while(t < Q[i].t) Make(++t);
```

```
    while(t > Q[i].t) Make(t--);
    Ans[Q[i].pos] = ans;
}
```

基本上没什么差距，会普通的约等于会带修改的。

复杂度证明

假设共有 N 个点， M 次操作， A 次查询，B 次修改， 块大小为 S。

对于 l 指针： - 块内移动，每次查询的最大代价为S，共有 A 次， 复杂度 SA。

对于 r 指针 - 随 l 指针移动，l 指针固定在一块中时，r 指针单调递增，最大代价为N，共有 $\frac{N}{S}$ 次， 复杂度 $\frac{N^2}{S}$

对于 t 指针 - 当l, r固定时，t单调递增，最大代价为 B次，共有$\frac{N^2}{S^2}$ 次， 复杂度 $\frac{BN^2}{S^2}$

推一推，由于题目中不会告诉你查询和修改的次数，所以 A 和 B 均视为 M。
即$O\left(SM+\frac{N^2}{S}+\frac{MN^2}{S^2}\right)$

这玩意的极值不好确定，取得精确结果比较困难。一般而言，在题目中 N 和 M 是同阶的，设 $S=N^x$ 则可得复杂度为$O\left(N^{x+1}+N^{2-x}+N^{3-2x}\right)$，我们希望其指数部分尽可能的小。即$\max\{x+1,2-x,3-2x\}$最小。解得$x=\frac{2}{3}$，即取$S=N^{\frac{2}{3}}$，此时的复杂度为$O\left(N^{\frac{5}{3}}\right)$。

例题

[国家集训队]数颜色

题意

有 n 个不同颜色的画笔，两种操作

第一种操作询问 $[L, R]$ 中有多少种不同颜色的画笔

第二种操作单独修改某一个画笔的颜色

题解

先考虑 (l, r, t) 前两维的变化，每次仅涉及一个颜色的增减。可以开一个桶存放当前每一种颜色的数量，当减少为 0 或者增加到 1 时修改答案。

考虑时间的修改，记录下修改的位置和颜色，如果不在当前区间内直接修改，如果在当前区间内，可以等价为一次区间增加和一次区间减小。

对了，这题卡常。需要写的常数很小才能过。

```cpp
#include <cmath>
#include <cstdio>
#include <iostream>
#include <algorithm>

const int N = 150000;
const int M = 1e6 + 100;
inline int read() {
    int q=0,w=1;char c=getchar();
    while(c<'0'||c>'9'){if(c=='-')w=-1;c=getchar();}
    while(c>='0'&&c<='9')q=(q<<1)+(q<<3)+(c^48),c=getchar();
    return w*q;
}
inline void swap(int &x,int &y){x=x^y;y=x^y;x=x^y;}

char s[5];
int n, m, i, j, l, r, t, ans, blk, cntQ, cntR;
int pos[N], val[N], cnt[M], a[N], Ans[N], tmp[N];

struct Que{
    int l, r, idl, idr, t, pos;
    inline bool operator < (const Que &b) const {
        if(idl ^ b.idl) return idl < b.idl;
        if(idr ^ b.idr) return idr < b.idr;
        return t < b.t;
    }
} Q[N];

int main() {
    n = read(); m = read(); blk=ceil(pow(n,0.75));
    for(i = 1;i <= n; ++i)
        a[i] = read();
    for(i = blk;i <= n; ++i) tmp[i] = tmp[i - blk] + 1;
    for(i = 1;i <= m; ++i) {
        scanf("%s", s);
        if(s[0]=='Q') {
            cntQ++;
            Q[cntQ].l = read();
            Q[cntQ].r = read();
            Q[cntQ].idl = tmp[Q[cntQ].l];
            Q[cntQ].idr = tmp[Q[cntQ].r];
            Q[cntQ].t = cntR;
            Q[cntQ].pos = cntQ;
        } else {
            cntR++;
            pos[cntR] = read();
            val[cntR] = read();
        }
    }
    std::sort(Q + 1, Q + 1 + cntQ);l = 1; r = 0; t = 0; ans = 0;
```

```
    for(i = 1;i <= cntQ; ++i) {
        int ql = Q[i].l, qr = Q[i].r, qt = Q[i].t;
        while(l < ql) ans -= !--cnt[a[l++]];
        while(l > ql) ans += !cnt[a[--l]]++;
        while(r < qr) ans += !cnt[a[++r]]++;
        while(r > qr) ans -= !--cnt[a[r--]];
        while(t < qt) {
            t++;
            if(l <= pos[t] && pos[t] <= r) ans -= !--cnt[a[pos[t]]] -
!cnt[val[t]]++;
            swap(a[pos[t]], val[t]);
        }
        while(t > qt) {
            if(l <= pos[t] && pos[t] <= r) ans -= !--cnt[a[pos[t]]] -
!cnt[val[t]]++;
            swap(a[pos[t]], val[t]);
 --t;
        }
        Ans[Q[i].pos] = ans;
    }
    for(i = 1;i <= cntQ; ++i)
        printf("%d\n", Ans[i]);
    return 0;
}
```