最基本需要掌握的如：线段树，RMQ，树状数组，分块……

时代在迫使人类进步

莫队算法

离线+暴力+分块

排序巧妙优化复杂度，带来NOIP前的最后一丝宁静。
几个活蹦乱跳的**指针的跳跃**次数，决定着莫队算法的优劣。

——摘自博客园

FZOI

信息学

莫队入门

（离线+分块 只能说相当优雅）

%%%%%%

Orz

Ps:只有普通莫队是莫队亲自提出的，其他的都是后人们自己改进的。而且莫队在论文里说了，CF的红名dalao基本上都会用普通莫队，只是没有人统一提出。

普通莫队

一个问题

#2877. 「SDOI2009」HH 的项链

时间限制:1 s 空间限制:256 MB

好评 差评 [0]

描述 提交 用户题解 来源 难度 自测 管理 代码查重 收藏题目 数据下载 附加文件 提交记录 统计

HH 有一串由各种漂亮的贝壳组成的项链。HH 相信不同的贝壳会带来好运，所以每次散步完后，他都会随意取出一段贝壳，思考它们所表达的含义。HH 不断地收集新的贝壳，因此，他的项链变得越来越长。有一天，他突然提出了一个问题：某一段贝壳中，包含了多少种不同的贝壳？这个问题很难回答……因为项链实在是太长了。于是，他只好求助睿智的你，来解决这个问题。

输入

第一行：一个整数N，表示项链的长度。

第二行：N 个整数，表示依次表示项链中贝壳的编号（编号为0 到1000000 之间的整数）。

第三行：一个整数M，表示HH 询问的个数。

接下来M 行：每行两个整数，L 和R（1 ≤ L ≤ R ≤ N），表示询问的区间。

输出

M 行，每行一个整数，依次表示询问对应的答案。

样例

样例输入1

```
6
1 2 3 4 3 5
3
1 2
3 5
2 6
```

样例输出1

```
2
2
4
```

提示

数据范围：

对于100%的数据，N <= 500000，M <= 500000。

题意：给定一个数量，询问某个区间内不同的数有多少个。
输入：第一行一个正整数 n，表示数列长度。
第二行n个正整数 ai。第三行一个整数m，表示HH 询问的个数。
接下来 m 行，每行两个整数 L,R，表示询问的区间。
输出：输出m行，每行一个整数，依次表示询问对应的答案。

暴力解法

最简单做法——

用一个cnt数组记录每个数值出现的次数，再暴力枚举 l 到 r 统计次数，最后再扫一遍cnt数组，统计cnt不为零的数值个数，输出答案即可。

设最大数值为s，那么这样做的复杂度为$O(m(n+s)) \backsim O(n^2)$，对于本题实在跑不起。

怎么优化?

每次枚举到一个数值num，增加出现次数时判断一下cnt_num是否为0，如果为0，则这个数值之前没有出现过，现在出现了，数值数当然要+1。反之在从区间中删除num后也判断一下cnt_num是否为0，如果为0数值总数-1。这样我们优化掉了一个O(ms)，但还是跑不起。

弄两个指针 l 、r ，每次询问不直接枚举，而是移动 l 、r 指针到询问的区间，直到 [l,r] 与询问区间重合。在统计答案时，我们也只在两个指针处加减cnt，这样就可以快速地统计答案啦

看哈具体步骤

为啥不是0 0？　要删除一个数值0，使其出现次数变成-1，容易有奇奇怪怪的错误

r　l

Q2

下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数值		1	2	4	2	7	7	3	1	4	2	6	5	1	2	3

Q1

看哈具体步骤

l 已经是第一个查询区间的左端点，无需移动。将 r 右移一位，发现新数值1

r
l

Q_2

下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数值		1	2	4	2	7	7	3	1	4	2	6	5	1	2	3

Q_1

总数：1

r 继续右移，发现新数值2：

l r

Q_2

下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数值		1	2	4	2	7	7	3	1	4	2	6	5	1	2	3

Q_1

总数：2

r 继续右移，发现新数值4：

总数：3

l r

Q2

下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数值		1	2	4	2	7	7	3	1	4	2	6	5	1	2	3

Q1

看哈具体步骤

r 继续右移，发现数值2,出现过，总数不变：

总数：3

l r

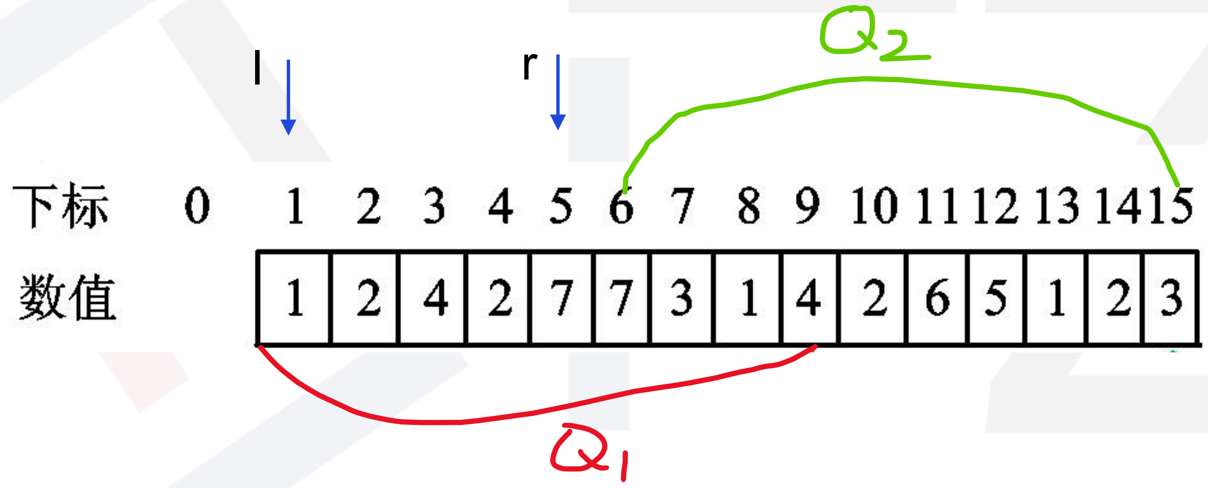
Q_2

下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数值		1	2	4	2	7	7	3	1	4	2	6	5	1	2	3

Q_1

看哈具体步骤

r 继续右移，发现新数值7：



总数：4

看哈具体步骤

l r

Q2

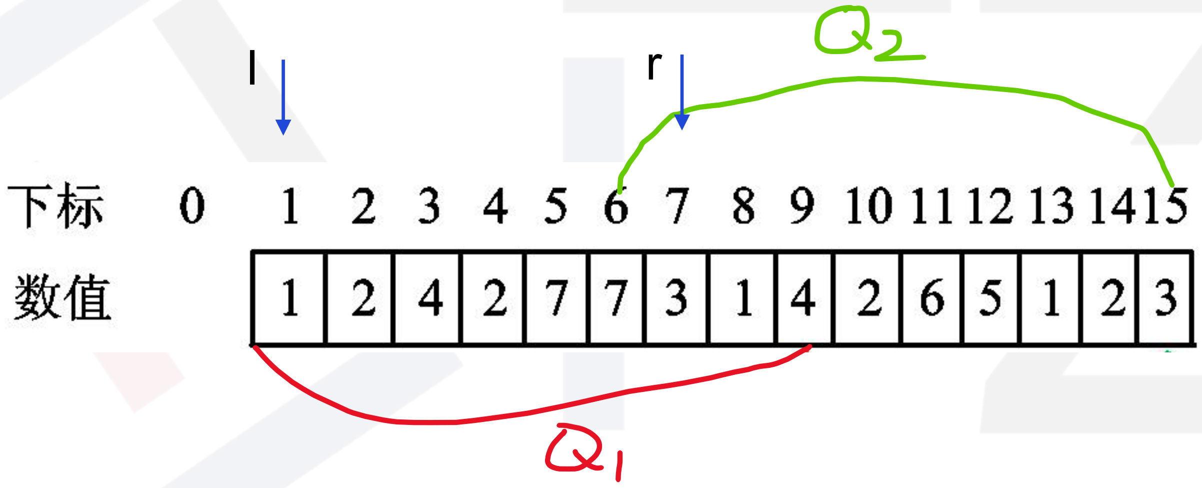
下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数值		1	2	4	2	7	7	3	1	4	2	6	5	1	2	3

Q1

总数：4

看哈具体步骤

r 继续右移，发现新数值3：



总数：5

看哈具体步骤

r 继续右移，1\4都出现过：

l r Q2

下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数值		1	2	4	2	7	7	3	1	4	2	6	5	1	2	3

Q1

总数：5

Q1区间所有数值统计完成，结果为5

看哈具体步骤

r 继续右移，1 4都出现过：

总数：5

l　　　　　　　　　　　r

Q_2

下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数值		1	2	4	2	7	7	3	1	4	2	6	5	1	2	3

Q_1

Q1区间所有数值统计完成，结果为5

看哈具体步骤

r 继续右移，1 4都出现过：

总数：5

l（下标 1） r（下标 9）

Q2

下标	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数值		1	2	4	2	7	7	3	1	4	2	6	5	1	2	3

Q1

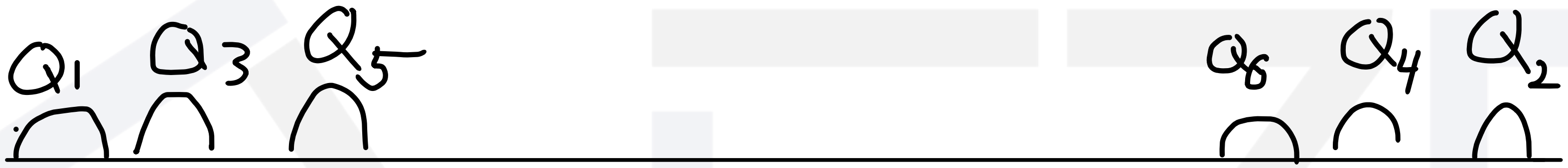
Q2怎么去推？自己做一下看看

代码也很好写

```
int aa[maxn], cnt[maxn], l = 1, r = 0, now = 0; //每个位置的数值、每个数值的计数器、左指针、右指针、当前统计结果（总数）
void add(int pos) {//添加一个数
    if(!cnt[aa[pos]]) ++now;//在区间中新出现，总数要+1
    ++cnt[aa[pos]];
}
void del(int pos) {//删除一个数
    --cnt[aa[pos]];
    if(!cnt[aa[pos]]) --now;//在区间中不再出现，总数要-1
}
void work() {//优化2主过程
    for(int i = 1; i <= q; ++i) {//对于每次询问
        int ql, qr;
        scanf("%d%d", &ql, &qr);//输入询问的区间
        while(l < ql) del(l++);//如左指针在查询区间左方，左指针向右移直到与查询区间左端点重合
        while(l > ql) add(--l);//如左指针在查询区间左端点右方，左指针左移
        while(r < qr) add(++r);//右指针在查询区间右端点左方，右指针右移
        while(r > qr) del(r--);//否则左移
        printf("%d\n", now);//输出统计结果
    }
}
```

但是莫队不是分块加暴力吗？分块呢？Tad吃了？

特殊情况



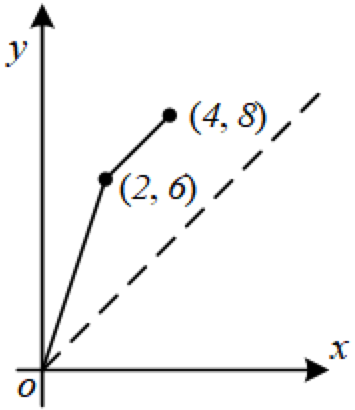
如果询问区间是这样，妥妥的炸掉

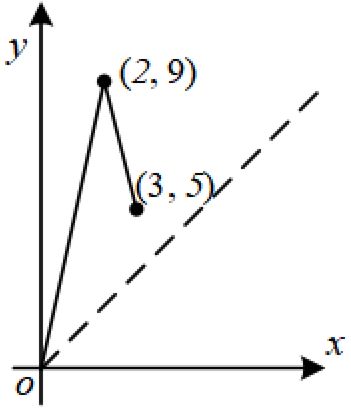
这个问题的暴力做法可以概括为：

1）m个询问对应m个区间，区间之间的转移，可以用L、R指针扫描，能以O(1)的复杂度从区间[L,R]移动到[L±1, R±1]。

2）把一个区间[L, R]看成平面上的一个坐标点(x, y)，L对应x，R对应y，那么区间的转移等同于平面上坐标点的转移，计算量等于坐标点之间的曼哈顿距离。注意，所有的坐标点(x, y)都满足x ≤ y，即所有的点都分布在上半平面上。

3）完成m个询问，等于从原点出发，用直线连接这m个点，形成一条“Hamilton路径”，路径的长度就是计算量。若能找到一条最短的路径，计算量就最少。





我们可以考虑把所有查询区间按左端点排序，从而使左指针最多移动O(n)次。
但这样的话右端点又是无序的，右指针又让整体复杂度打回原形。看上去，这个复杂度已经不能再优化了。

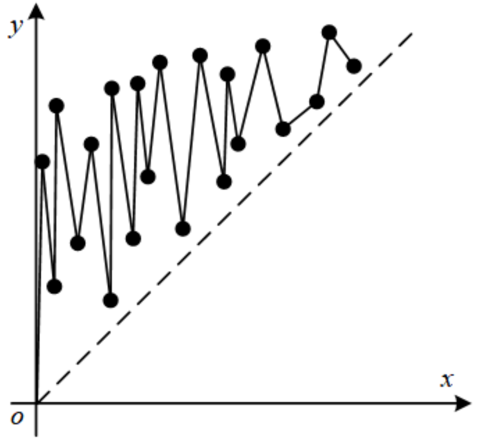
莫队算法

分块这不就来了嘛！

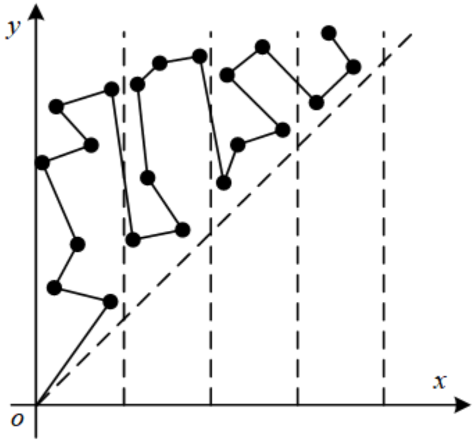
莫队算法把排序做了简单的修改，就把暴力法的复杂度从$O(mn)$提高到$O(n\sqrt{n})$。

（1）暴力法的排序：把查询的区间按左端点排序，如果左端点相同，再按右端点排序。

（2）莫队算法的排序：把数组分块（分成 $\sqrt{n}$ 块），然后把查询的区间按**左端点所在块的序号排序**，如果左端点的块相同，再按**右端点排序**（注意不是按右端点所在的块排序）。



暴力

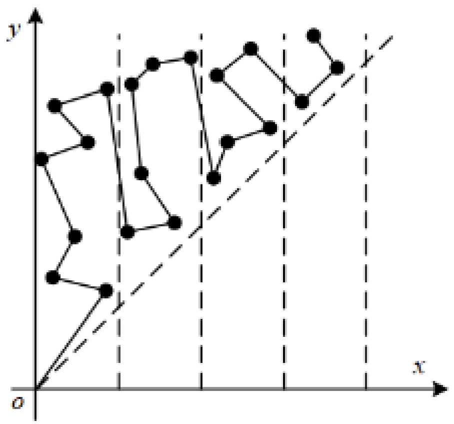


莫队

它把x轴分成多个区（分块），每个区内的点按y坐标排序，在区内沿x方向来回往复，此时震荡幅度被限制在区内（横向震荡，幅度$O(\sqrt{n})$），形成了一条比较短的路径，从而实现了较好的复杂度。

莫队算法

时间复杂度



（1）x方向的复杂度。在一个区块内，沿着x方向一次移动最多 $\sqrt{n}$ ，所有区块共有m次移动，总复杂度$O(m\sqrt{(n)})$。

（2）y方向的复杂度。在每个区块内，沿着y方向单向移动，整个区块的y方向长度是n，有 $\sqrt{n}$ 个区块，总复杂度$O(n\sqrt{(n)})$。

两者相加，总复杂度$O(m\sqrt{(n)}+n\sqrt{(n)})$。

why?

莫队算法把排序做了简单的修改，就把暴力法的复杂度从$O(mn)$提高到$O(n\sqrt{n})$。

（1）暴力法的排序：把查询的区间按左端点排序，如果左端点相同，再按右端点排序。

（2）莫队算法的排序：把数组分块（分成 $\sqrt{n}$ 块），然后把查询的区间按**左端点所在块的序号排序**，如果左端点的块相同，再按**右端点排序**（注意不是按右端点所在的块排序）。

怎么分块

我们设块长度为 S，那么对于任意多个在同一块内的询问，挪动的距离就是 n，一共 $\frac{n}{S}$ 个块，移动的总次数就是 $\frac{n^2}{S}$，移动可能跨越块，所以还要加上一个 mS 的复杂度，总复杂度为 $O\left(\frac{n^2}{S}+mS\right)$，我们要让这个值尽量小，那么就要将这两个项尽量相等，发现 S 取 $\frac{n}{\sqrt{m}}$ 是最优的，此时复杂度为 $O\left(\frac{n^2}{\frac{n}{\sqrt{m}}}+m\left(\frac{n}{\sqrt{m}}\right)\right)=O(n\sqrt{m})$。

事实上，如果块长度的设定不准确，则莫队的时间复杂度会受到很大影响。例如，如果 m 与 $\sqrt{n}$ 同阶，并且块长误设为 $\sqrt{n}$，则可以很容易构造出一组数据使其时间复杂度为 $O(n\sqrt{n})$ 而不是正确的 $O(n)$。

不要盲目的划分长度

只是排序不同

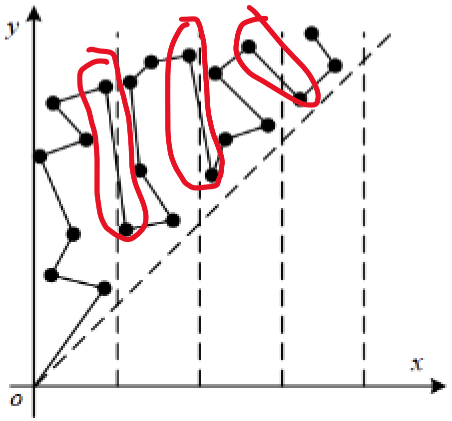
```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e6;
struct node{          //离线记录查询操作
  int L, R, k;        //k: 查询操作的原始顺序
}q[maxn];
int pos[maxn];
int ans[maxn];
int cnt[maxn];        //cnt[i]: 统计数字i出现了多少次
int a[maxn];
bool cmp(node a, node b){
    //按块排序，就是莫队算法：
    if(pos[a.L] != pos[b.L])     //按L所在的块排序，如果块相等，再按R排序
        return pos[a.L] < pos[b.L];

    return a.R < b.R;
        /*如果不按块排序，而是直接L、R排序，就是普通暴力法：
        if(a.L==b.L)  return a.R < b.R;
        return a.L < b.L;   */
}
int ANS = 0;
void add(int x){      //扩大区间时（L左移或R右移），增加数x出现的次数
    cnt[a[x]]++;
    if(cnt[a[x]]==1)  ANS++;    //这个元素第1次出现
}
void del(int x){      //缩小区间时（L右移或R左移），减少数x出现的次数
    cnt[a[x]]--;
    if(cnt[a[x]]==0)  ANS--;    //这个元素消失了
}
```

```
int main(){
    int n; scanf("%d",&n);
    int block = sqrt(n);              //每块的大小
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);            //读第i个元素
        pos[i]=(i-1)/block + 1;       //第i个元素所在的块
    }
    int m; scanf("%d",&m);
    for(int i=1;i<=m;i++){            //读取所有m个查询，离线处理
        scanf("%d%d",&q[i].L, &q[i].R);
        q[i].k = i;                   //记录查询的原始顺序
    }
    sort(q+1, q+1+m, cmp);            //对所有查询排序
    int L=1, R=0;                     //左右指针的初始值。思考为什么？
    for(int i=1;i<=m;i++){
        while(L<q[i].L)  del(L++);    //{del(L); L++;}  //缩小区间：L右移
        while(R>q[i].R)  del(R--);    //{del(R); R--;}  //缩小区间：R左移
        while(L>q[i].L)  add(--L);    //{L--; add(L);}  //扩大区间：L左移
        while(R<q[i].R)  add(++R);    //{R++; add(R);}  //扩大区间：R右移
        ans[q[i].k] = ANS;
    }
    for(int i=1;i<=m;i++)   printf("%d\n",ans[i]);   //按原顺序打印结果
    return 0;
}
```

看一组数据

块	右端点
1	1
2	100
3	1
4	100

指针怎么动的?



进一步优化:
奇偶性排序，让奇数块和偶数块的排序相反。
例如左端点L都在奇数块，则对R从大到小排序;
若L在偶数块，则对R从小到大排序（反过来也可以:
奇数块从小到大，偶数块从大到小）。

```
bool cmp(node a, node b){
    //按块排序，就是莫队算法:
    if(pos[a.L] != pos[b.L])         //按L所在的块排序，如果块相等，再按R排序
        return pos[a.L] < pos[b.L];

    if(pos[a.L] & 1)    return a.R > b.R; //奇偶性优化，如果删除这一句，性能差一点

    return a.R < b.R;
}
```

适用范围

离线

多次区间查询，查询一些线段树之类数据结构做不了的事情（区间第一个未出现的自然数、区间不同的数个数……）。

还有一些特性，比如：区间扩大或缩小一个位置要比较快，要能分块。

小z的袜子

#127. 「BZOJ2038」小 Z 的袜子

时间限制:1 s　空间限制:128 MB

1817

好评　差评　[+5]

描述　提交　用户题解　来源　难度　自测　管理　代码查重　收藏题目　数据下载　附加文件　提交记录　统计

题目描述

作为一个生活散漫的人，小Z每天早上都要耗费很久从一堆五颜六色的袜子中找出一双来穿。终于有一天，小Z再也无法忍受这恼人的找袜子过程，于是他决定听天由命……

具体来说，小Z把这N只袜子从1到N编号，然后从编号L到R(L 尽管小Z并不在意两只袜子是不是完整的一双，甚至不在意两只袜子是否一左一右，他却很在意袜子的颜色，毕竟穿两只不同色的袜子会很尴尬。

你的任务便是告诉小Z，他有多大的概率抽到两只颜色相同的袜子。当然，小Z希望这个概率尽量高，所以他可能会询问多个(L, R)以方便自己选择。

输入格式

输入文件第一行包含两个正整数N和M。N为袜子的数量，M为小Z所提的询问的数量。接下来一行包含N个正整数C_i，其中C_i表示第i只袜子的颜色，相同的颜色用相同的数字表示。再接下来M行，每行两个正整数L，R表示一个询问。

输出格式

包含M行，对于每个询问在一行中输出分数A/B表示从该询问的区间$[L, R]$中随机抽出两只袜子颜色相同的概率。若该概率为0则输出$0/1$，否则输出的A/B必须为最简分数。（详见样例）

样例输入

```
6 4
1 2 3 3 3 2
2 6
1 3
3 5
1 6
```

样例输出

```
2/5
0/1
1/1
4/15
```

样例解释

询问1：共$\binom{5}{2} = 10$种可能，其中抽出两个2有1种可能，抽出两个3有3种可能，概率为$(1+3)/10 = 4/10 = 2/5$。

询问2：共$\binom{3}{2} = 3$种可能，无法抽到颜色相同的袜子，概率为$0/3 = 0/1$。

询问3：共$\binom{3}{2} = 3$种可能，均为抽出两个3，概率为$3/3 = 1/1$。

注：上述$\binom{a}{b}$表示组合数，组合数$\binom{a}{b}$等价于在a个不同的物品中选取b个的选取方案数。

带修莫队（可持久化）

想要修改怎么办?

单点修改，区间查询

墨墨购买了一套N支彩色画笔（其中有些颜色可能相同），摆成一排，你需要回答墨墨的提问。墨墨会像你发布如下指令：

1、 Q L R代表询问你从第L支画笔到第R支画笔中共有几种不同颜色的画笔。

2、 R P Col 把第P支画笔替换为颜色Col。为了满足墨墨的要求，你知道你需要干什么了吗?

输入

第1行两个整数N，M，分别代表初始画笔的数量以及墨墨会做的事情的个数。

第2行N个整数，分别代表初始画笔排中第i支画笔的颜色。第3行到第2+M行，每行分别代表墨墨会做的一件事情，格式见题干部分。

输出

对于每一个Query的询问，你需要在对应的行中给出一个数字，代表第L支画笔到第R支画笔中共有几种不同颜色的画笔。

样例

样例输入1

```
6 5
1 2 3 4 5 5
Q 1 4
Q 2 6
R 1 2
Q 1 4
Q 2 6
```

样例输出1

```
4
4
3
4
```

提示

对于100%的数据，N≤10000，M≤10000，修改操作不多于1000次，所有的输入数据中出现的所有整数均大于等于1且不超过10^6。

想想前面讲的可持久化数据结构

加个时间维度！

还是对询问进行排序，每个询问除了左端点和右端点还要记录这次询问是在第几次修改之后（时间），

1. 以左端点所在块为第一关键字，
2. 以右端点所在块为第二关键字，
3. 以时间为第三关键字进行排序。

单点修改

西南大学附属中学 High School Affiliated to Southwest University

考虑**普通莫队**：

每次扩大区间时，每加入一个数字，则统计它已经出现的次数，如果加入前这种数字出现次数为 0，则说明这是一种新的数字，答案 +1。然后这种数字的出现次数 +1。

每次减小区间时，每删除一个数字，则统计它删除后的出现次数，如果删除后这种数字出现次数为 0，则说明这种数字已经从当前的区间内删光了，也就是当前区间减少了一种颜色，答案 -1。然后这种数字的出现次数 -1。

考虑**修改**：

单点修改，把某一位的数字修改掉。假如我们是从一个经历修改次数为 i 的询问转移到一个经历修改次数为 j 的询问上，且 i<j 的话，我们就需要把第 i+1 个到第 j 个修改强行加上。

假如 j<i 的话，则需要把第 i 个到第 j+1 个修改强行还原。

怎么强行加上一个修改呢？假设一个修改是修改第 pos 个位置上的颜色，原本 pos 上的颜色为 a，修改后颜色为 b，还假设当前莫队的区间扩展到了 [l,r]。

加上这个修改：我们首先判断 pos 是否在区间 [l,r] 内。如果是的话，我们等于是从区间中删掉颜色 a，加上颜色 b，并且当前颜色序列的第 pos 项的颜色改成 b。如果不在区间 [l,r] 内的话，我们就直接修改当前颜色序列的第 pos 项为 b。

还原这个修改：等于加上一个修改第 pos 项、把颜色 b 改成颜色 a 的修改。

几何解释

带修改的莫队(L,R,t)对应立体空间的$(x\ ,y\ ,z\ )$。每个查询对应立体空间的一个点，那么从一个查询到另一个查询，就是从一个点(x_1,y_1,z_1)到另一个点(x_2,y_2,z_2)。计算复杂度仍然是两点之间的曼哈顿距离。

模仿基础莫队的分块思路。定义带修改莫队的排序，按以下步骤执行：

（1）按左端点L排序。若左端点L在同一个块，执行（2）。L对应x xx轴。

（2）按右端点R排序。若右端点R在同一个块，执行（3）。R对应y yy轴。

（3）按时间t排序。t对应z zz轴。

左端点L所在的块是第1查询关键字，右端点R所在的块是第2关键字，时间t是第3关键字。

x方向和y方向的分块，把x xx-y yy平面分成了方格，代表查询的点在方格内、方格间移动。

怎么分块?

x方向和y方向的分块，把 x-y 平面分成了方格，代表查询的点在方格内、方格间移动。根据带修改莫队的几何意义，计算算法的复杂度。这里先不采用 的分块方法，而是设一个分块的大小是B，共有n/B个分块。计算x 、 y 、 z 三个方向上的复杂度：

（1）x 方向的复杂度（左端点指针L）。在一个区块内，沿着x 方向一次最多移动B，所有的区块共有m次移动，总计算量 = mB。

（2）y 方向的复杂度（右端点指针R）。在一个区块内，沿着y 方向一次最多移动B，所有的区块共有m次移动，总计算量 = mB。

（3）z 方向的复杂度（时间t）。每个被x 和y 区块限制的方格内，沿着z 方向单向移动，最多移动m次，共n^2/B^2个方格，总计算量 = $(mn^2)/B^2$

三者相加，总计算量 = $mB + mB + (mn^2)/B^2$ 。**当$B = n^{\frac{2}{3}}$**时有较好的复杂度$O\left(mn^{\frac{2}{3}}\right)$。

作为对照，如果分块$B = \sqrt{(n)}$，复杂度是$O(mn)$，退化成了暴力法的复杂度。