[编程题] 用户喜好

时间限制：3秒

空间限制：262144K

为了不断优化推荐效果，今日头条每天要存储和处理海量数据。假设有这样一种场景：我们对用户按照它们的注册时间先后来标号，对于一类文章，每个用户都有不同的喜好值，我们会想知道某一段时间内注册的用户（标号相连的一批用户）中，有多少用户对这类文章喜好值为k。因为一些特殊的原因，不会出现一个查询的用户区间完全覆盖另一个查询的用户区间(不存在L1<=L2<=R2<=R1)。

输入描述:

输入： 第1行为n代表用户的个数 第2行为n个整数，第i个代表用户标号为i的用户对某类文章的喜好度 第3行为一个正整数q代表查询的组数 第4行到第（3+q）行，每行包含3个整数l,r,k代表一组查询，即标号为l<=i<=r的用户中对这类文章喜好值为k的用户的个数。 数据范围n <= 300000,q<=300000 k是整型

输出描述:

输出：一共q行，每行一个整数代表喜好值为k的用户的个数

输入例子1:

5

1 2 3 3 5

3

1 2 1

2 4 5

3 5 3

输出例子1:

1

0

2

例子说明1:

样例解释:

有5个用户，喜好值为分别为1、2、3、3、5，

第一组询问对于标号[1,2]的用户喜好值为1的用户的个数是1

第二组询问对于标号[2,4]的用户喜好值为5的用户的个数是0

第三组询问对于标号[3,5]的用户喜好值为3的用户的个数是2

[编程题] 手串

时间限制：1秒

空间限制：65536K

作为一个手串艺人，有金主向你订购了一条包含n个杂色串珠的手串——每个串珠要么无色，要么涂了若干种颜色。为了使手串的色彩看起来不那么单调，金主要求，手串上的任意一种颜色（不包含无色），在任意连续的m个串珠里至多出现一次（注意这里手串是一个环形）。手串上的颜色一共有c种。现在按顺时针序告诉你n个串珠的手串上，每个串珠用所包含的颜色分别有哪些。请你判断该手串上有多少种颜色不符合要求。即询问有多少种颜色在任意连续m个串珠中出现了至少两次。

输入描述:

第一行输入n，m，c三个数，用空格隔开。(1 <= n <= 10000, 1 <= m <= 1000, 1 <= c <= 50) 接下来n行每行的第一个数num\_i(0 <= num\_i <= c)表示第i颗珠子有多少种颜色。接下来依次读入num\_i个数字，每个数字x表示第i颗柱子上包含第x种颜色(1 <= x <= c)

输出描述:

一个非负整数，表示该手链上有多少种颜色不符需求。

输入例子1:

5 2 3

3 1 2 3

0

2 2 3

1 2

1 3

输出例子1:

2

例子说明1:

第一种颜色出现在第1颗串珠，与规则无冲突。

第二种颜色分别出现在第 1，3，4颗串珠，第3颗与第4颗串珠相邻，所以不合要求。

第三种颜色分别出现在第1，3，5颗串珠，第5颗串珠的下一个是第1颗，所以不合要求。

总计有2种颜色的分布是有问题的。

这里第2颗串珠是透明的。

[编程题] 字母交换

时间限制：1秒

空间限制：32768K

【编码题】字符串S由小写字母构成，长度为n。定义一种操作，每次都可以挑选字符串中任意的两个相邻字母进行交换。询问在至多交换m次之后，字符串中最多有多少个连续的位置上的字母相同？

输入描述:

第一行为一个字符串S与一个非负整数m。(1 <= |S| <= 1000, 1 <= m <= 1000000)

输出描述:

一个非负整数，表示操作之后，连续最长的相同字母数量。

输入例子1:

abcbaa 2

输出例子1:

2

例子说明1:

使2个字母a连续出现，至少需要3次操作。即把第1个位置上的a移动到第4个位置。

所以在至多操作2次的情况下，最多只能使2个b或2个a连续出现。

[问答题]

题目描述

以下函数使用二分查找搜索一个增序的数组，当有多个元素值与目标元素相等时，返回最后一个元素的下标，目标元素不存在时返回-1。请指出程序代码中错误或不符最佳实践的地方（问题不止一处，请尽量找出所有你认为有问题的地方）

int BinarySearchMax(const std::vector<int>& data, int target)

{

int left = 0;

int right = data.size();

while (left < right) {

int mid = (left + right) / 2;

if (data[mid] <= target)

left = mid + 1;

else

right = mid - 1;

}

if (data[right] == target)

return right;

return -1;

}

[问答题]

题目描述

【设计题】今日头条会根据用户的浏览行为、内容偏好等信息，为每个用户抽象出一个标签化的用户画像，用于内容推荐。用户画像的存储、高并发访问，是推荐系统的重要环节之一。现在请你给出一个用户画像存储、访问方案，设计的时候请考虑一下几个方面：

用户画像如何存储

如何保证在线高并发、低延迟地访问

机器宕机、负载均衡问题

如果用户增长很快，在你的方案下，该如何做扩容

1. #include <bits/stdc++.h>  
using namespace std;  
const int maxn = 300010;  
unordered\_map <int,int> mp;  
struct node{  
    int l,r,k,id;  
}Q[maxn];  
int n,m;  
int pos[maxn];  
int a[maxn],Ans[maxn];  
bool cmp(const node &a, const node &b){  
    if(pos[a.l]==pos[b.l]) return a.r<b.r;  
    return pos[a.l]<pos[b.l];  
}  
  
int main(){  
    while(~scanf("%d",&n)){  
        mp.clear();  
        int sz=sqrt(n);  
        for(int i=1;i<=n;i++){  
            scanf("%d",&a[i]);  
            pos[i]=i/sz;  
        }  
        scanf("%d",&m);  
        for(int i=1;i<=m;i++){  
            scanf("%d%d%d",&Q[i].l,&Q[i].r,&Q[i].k);  
            Q[i].id=i;  
        }  
        sort(Q+1,Q+m+1,cmp);  
        int L=1,R=0;  
        for(int i=1;i<=m;i++){  
            while(L<Q[i].l){  
                mp[a[L]]--;  
                ++L;  
            }  
            while(L>Q[i].l){  
                --L;  
                mp[a[L]]++;  
            }  
            while(R<Q[i].r){  
                ++R;  
                mp[a[R]]++;  
            }  
            while(R>Q[i].r){  
                mp[a[R]]--;  
                R--;  
            }  
            Ans[Q[i].id] = mp[Q[i].k];  
        }  
        for(int i=1;i<=m;i++) printf("%d\n", Ans[i]);  
    }  
    return 0;  
}

//模拟题没啥说的。

#include <bits/stdc++.h>

using namespace std;  
const int maxn = 110;  
vector <int> v[maxn];  
int n, m, c;  
int find\_key(int color, int key){  
    for(int i=0;i<v[color].size();i++){  
        if(v[color][i]==key) return 1;  
    }  
    return 0;  
}  
int main(){  
    while(~scanf("%d%d%d",&n,&m,&c)){  
        for(int i=0;i<maxn;i++) v[i].clear();  
        for(int i=1;i<=n;i++){  
            int num,x;  
            scanf("%d",&num);  
            while(num--){  
                scanf("%d",&x);  
                v[x].push\_back(i);  
            }  
        }  
        for(int i=1;i<=c;i++){  
            if(v[i].empty()) continue;  
            vector <int> temp;  
            for(int j=0; j<v[i].size(); j++) temp.push\_back(v[i][j]);  
            for(int j=0;j<temp.size();j++) v[i].push\_back(temp[j]+n);  
        }  
//        for(int i=1;i<=c;i++){  
//            if(v[i].empty()) continue;  
//            cout<<v[i].size()<<endl;  
//            for(int j=0;j<v[i].size();j++) cout<<v[i][j]<<" ";  
//            cout<<endl;  
//        }  
        int ans = 0;  
        for(int i=1;i<=c;i++){  
            if(v[i].empty()) continue;  
            int l=1,r=1,sum=find\_key(i,1);  
            for(;l+m<=n\*2;l++){  
                while(r-l+1<m){  
                    ++r;  
                    sum+=find\_key(i,r);  
                }  
                if(sum>1){  
                    ans++;  
                    break;  
                }  
                sum-=find\_key(i,l);  
            }  
        }  
        printf("%d\n", ans);  
    }  
}