2020牛客暑期多校训练营（第二场）

比赛网址

训练结果

- 时间:2020-7-13 12:00~17:00
- rank:145/1159
- 完成情况：4/8/11

题解

题目名字

题意

题解

H.Happy Triangle

题意

一个multiset□支持如下操作：

- 插入一个数x
- 从中删除一个数x□如果有重复的，只删除一个）
- 给定x□问集合中是否存在两个数a,b□使得a,b,x组成一个非退化三角形。

题解

补题 by fyh

询问即问是否存在两个数a,b□使得$|a-b|<x<a+b$□这个很明显是一个区间覆盖问题，但是因为集合中两两组成的区间是n^2级别的，考虑减少区间□$a>b>c$,其中(a,b)与(a,c)组成的开区间分别是$(a-b,a+b),(a-c,a+c)$□后者是完全被前者包含的，所以对于每一个数，只需要找他的前驱，用线段树维护一下这n个区间的并即可。

```
#include<bits/stdc++.h>
using namespace std;
```

```
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=200010;
int n,q,tp[maxn],a[maxn],b[maxn],tag[maxn<<2],w[maxn<<2];
multiset <int> S;
multiset <int>::iterator it,it2;
void pushdown(int L,int R,int o)
{
    int mid=L+R>>1,lo=o<<1,ro=lo|1;
    tag[lo]+=tag[o];tag[ro]+=tag[o];
    w[lo]+=tag[o];w[ro]+=tag[o];
    tag[o]=0;
}
void update(int L,int R,int o,int ql,int qr,int v)
{
    if(L==ql && R==qr)
    {
        w[o]+=v;
        tag[o]+=v;
        return;
    }
    pushdown(L,R,o);
    int mid=L+R>>1,lo=o<<1,ro=lo|1;
    if(qr<=mid)update(L,mid,lo,ql,qr,v);
    else if(ql>mid)update(mid+1,R,ro,ql,qr,v);
    else update(L,mid,lo,ql,mid,v),update(mid+1,R,ro,mid+1,qr,v);
}
int query(int L,int R,int o,int qx)
{
    if(L==R)return w[o];
    pushdown(L,R,o);
    int mid=L+R>>1,lo=o<<1,ro=lo|1;
    if(qx<=mid)return query(L,mid,lo,qx);
    else return query(mid+1,R,ro,qx);
}
void insert(int A)
{
    int pos=lower_bound(b+1,b+n+1,A)-b,l,r;
    it=S.lower_bound(A);
    if(it!=S.end() && it!=S.begin())
```

```
    {
        it--;it2=it;it++;
        l=lower_bound(b+1,b+n+1,(*it)-*(it2))-b;
        r=lower_bound(b+1,b+n+1,(*it)+*(it2))-b-1;
        if(*it-*it2==b[l])l++;
        update(1,n,1,l,r,-1);
    }
    if(it!=S.end())
    {
        l=lower_bound(b+1,b+n+1,*it-A)-b;
        r=lower_bound(b+1,b+n+1,*it+A)-b-1;
        if(*it-A==b[l])l++;
        update(1,n,1,l,r,1);
    }
    if(it!=S.begin())
    {
        it--;it2=it;it++;
        l=lower_bound(b+1,b+n+1,A-*it2)-b;
        r=lower_bound(b+1,b+n+1,*it2+A)-b-1;
        if(A-*it2==b[l])l++;
        update(1,n,1,l,r,1);
    }
    S.insert(A);
}
void delt(int A)
{
    int pos=lower_bound(b+1,b+n+1,A)-b,l,r;
    it=S.find(A);
    S.erase(it);
    it=S.lower_bound(A);
    if(it!=S.end() && it!=S.begin())
    {
        it--;it2=it;it++;
        l=lower_bound(b+1,b+n+1,(*it)-*(it2))-b;
        r=lower_bound(b+1,b+n+1,(*it)+*(it2))-b-1;
        if(*it-*it2==b[l])l++;
        update(1,n,1,l,r,1);
    }
    if(it!=S.end())
    {
        l=lower_bound(b+1,b+n+1,*it-A)-b;
        r=lower_bound(b+1,b+n+1,*it+A)-b-1;
        if(*it-A==b[l])l++;
        update(1,n,1,l,r,-1);
    }
    if(it!=S.begin())
    {
        it--;it2=it;it++;
        l=lower_bound(b+1,b+n+1,A-*it2)-b;
        r=lower_bound(b+1,b+n+1,*it2+A)-b-1;
        if(A-*it2==b[l])l++;
```

```
        update(1,n,1,l,r,-1);
    }
}
void ask(int x)
{
    int pos=lower_bound(b+1,b+n+1,x)-b;
    if(query(1,n,1,pos))puts("Yes");
    else puts("No");
}
int main()
{
    q=read();
    for(int i=1;i<=q;i++)tp[i]=read(),a[i]=b[i]=read();
    sort(b+1,b+q+1);
    n=unique(b+1,b+q+1)-b-1;
    b[n+1]=1e9;
    for(int i=1;i<=q;i++)
    {
        if(tp[i]==1)insert(a[i]);
        else if(tp[i]==2)delt(a[i]);
        else ask(a[i]);
    }
    return 0;
}
```

题目名字

题意

题解

J.Just Shuffle

题意

一个排列初始时是$1,2,\ldots,n$□存在某种置换p,使得排列在置换k次后的排列为$A_1,\ldots,A_n$□求置换p□

题解

本题的一大特性是k是质数，也就是k模任何数都非0。

置换其实就是若干个循环。每个循环都是独立的，现考虑某个长度为$size$的环，置换k次的结果是等效于置换为$k\%size$的结果。设$k\%size=m$则我们当前得到的环其实是走m步的结果，我们要得到走1步的结果，便可以考虑在这个环上每步都好几倍，最后等效为走一步，即解$m*k\%size=1$的模方程的x至此，我们成功构造出了原置换。

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=100010;
int n,k,to[maxn],col[maxn],cnt,size[maxn],ans[maxn];
bool vis[maxn];
vector <int> V[maxn];
int main()
{
    n=read();k=read();
    for(int i=1;i<=n;i++)to[i]=read();
    for(int i=1;i<=n;i++)
        if(!vis[i])
        {
            col[i]=++cnt;
            V[cnt].push_back(i);
            int now=i;
            while(!vis[to[now]])
                V[cnt].push_back(to[now]),
                vis[to[now]]=1,col[to[now]]=cnt,
                now=to[now],size[cnt]++;
        }
    for(int i=1;i<=cnt;i++)
    {
        if(size[i]==1)ans[V[i][0]]=V[i][0];
        else
        {
            int m=k%size[i],x=0;
            while(x*m%size[i]!=1)x++;
            for(int j=0;j<V[i].size();j++)
            {
                int now=V[i][j];
                ans[V[i][j]]=V[i][(j+x)%size[i]];
            }
```

```
            }
        }
        for(int i=1;i<n;i++)printf("%d ",ans[i]);
        printf("%d\n",ans[n]);
        return 0;
}
```

训练实况

开场 发现**D**很简单
12:08 hxm 过**D** wxg想出**B**题做法 fyh开写**B**
12:08~12:50 fyh狂wa**B** wxg hxm想出**C** hxm开写**C**
13:33 hxm过C 在想**B**的错误和**F**，尝试用分数处理精度问题 改为wxg写**B**
13:33~14:30 **B**卡常 又wa又T hxm想出**F** 开写
15□04 hxm过**F** wxg继续尝试**B**，放弃 □wxg开想**G**
15:04~16:00 □□
16:00 看J题过得人多 fyh想**J**
16:36 fyh 过**J** wxg写**G**,未知原因段错误，结束。

训练总结

因为很少人过，没有读**I**
K题因为过的人少没有深想
H没有仔细想
B陷入无解，纠结太久
总结：计算几何最后再写 对榜的利用价值??