

# CodeChef February Challenge 2021

[比赛链接](#)

## Prime Game

### 题意

给定一个数  $A$  并且  $A$  的初始值为  $X$

接下来两个人轮流操作，每次操作选择一个不超过  $N$  且质因子种数不超过  $Y$  的正整数  $D$  得到  $A-D$

先找到  $0$  的玩家获胜。给定  $T$  组询问，每个询问给定  $X$  询问先手玩家是否可以必胜。

### 题解

首先将所有素数从小到大排列，记为  $p_1, p_2, \dots$  同时设  $B = \prod_{i=1}^{Y+1} p_i$

记状态一表示  $B \mid A$  状态二表示  $B \nmid A$

显然如果当前处于状态一，则下一步一定进入状态二，因为  $B \nmid D$

如果当前处于状态二，则可以取  $D \equiv A \pmod{B}$  则  $B \mid A-D$  且  $B \nmid D$  所以  $D$  至多有  $Y$  种素因子，即  $D$  合法。

于是如果  $A$  一开始是状态一，则每次行动玩家一一定进入状态二，而玩家二一定有办法回到状态一。

由于状态二中不存在  $0$ ，所以玩家一必败。如果  $A$  一开始是状态二，则玩家一像上一种情况的玩家二一样操作即可，此时玩家一必胜。

于是线性筛预处理，然后判定  $X$  是否不超过  $p_y$  即可，时间复杂度  $O(X+T)$

## Multiple Games

### 题意

给定严格递增的正整数序列  $A_1, A_2, \dots, A_n$  保证  $A_i + A_1 \geq A_{i+1}$  一开始由我方选定一个  $G$  使得  $0 \leq G \leq M$

接下来  $q$  场游戏，每场游戏我方先手，且一开始有  $G$  个石头。第  $i$  场游戏每次可以拿  $\{A_1, A_2, \dots, A_r\}$  个石头。

问必胜场次最多有几场。

## 题解

首先给出两个博弈游戏等价的定义：对同一个状态（本题为当前石头数），两个博弈游戏要么都是必胜状态，要么都是必败状态。

另外假设每次可以拿的石头为  $[l, r]$  个，则必胜状态为  $G \bmod (l+r) \geq l$

接下来给出两个条件：

1. 每次可以拿  $S = \{a_1, a_2, \dots, a_k\}$  个石头的游戏等价于每次可以拿  $[\min S, \max S]$  个石头的游戏。
2. 对任意  $a_i, a_j \in S$ ，若  $a_i + \min S < a_j$ ，则存在  $a_k \in S$  使得  $a_i < a_k < a_j$ 。

下面证明这两个条件等价。首先不妨令  $a_1 < a_2 < \dots < a_n$

当条件一成立时，假设存在  $a_i + a_1 < a_j$  且不存在  $a_k \in S$  使得  $a_i < a_k < a_j$  的情况。

于是有  $j = i+1$  即  $a_i + a_1 < a_{i+1}$  取  $G \bmod (a_1 + a_k) = a_1 + a_i$  根据条件一  $G$  是必胜状态。

于是如果选取  $a_1 \sim a_i$  则  $a_1 \leq G \bmod (a_1 + a_k) \leq a_i$  根据条件一  $G$  是必胜状态。

如果选取  $a_{i+1} \sim a_k$  则  $G \bmod (a_1 + a_k) \geq 2a_1 + a_i$  根据条件一  $G$  是必胜状态。

于是  $G$  是必败状态，矛盾。于是充分性证毕。

当条件二成立时，首先考虑  $0 \leq G \leq a_1 + a_k$  易知  $0 \leq G \leq a_1$  是必败状态。

当  $a_i \leq G \leq a_{i+1}$  时，取  $a_i$  个石头，根据条件二，有  $a_i + a_1 \geq a_{i+1}$  于是  $G' = G - a_i \leq a_1$  是必败状态。

于是  $a_1 \leq G \leq a_k$  是必胜状态。当  $a_k \leq G \leq a_1 + a_k$  时取  $a_k$  个石头有  $G \leq a_i$  于是  $G$  也是必胜状态。

于是  $0 \leq G \leq a_1 + a_k$  时必胜状态为  $a_1 \leq G \leq a_1 + a_k$

数学归纳法设  $k(a_1 + a_k) \leq G \leq (k+1)(a_1 + a_k)$  满足条件一。

当  $(k+1)(a_1 + a_k) \leq G \leq (k+1)(a_1 + a_k) + a_1$  时，任意取石头  $a_1 \sim a_k$

发现总有  $k(a_1 + a_k) + a_1 \leq G \leq (k+1)(a_1 + a_k)$  全是必胜状态，于是  $G$  是必败状态。

当  $(k+1)(a_1 + a_k) + a_1 \leq G \leq (k+2)(a_1 + a_k)$  类比  $a_1 \leq G \leq a_1 + a_k$  的取法即可到达必败状态，于是  $G$  是必胜状态。必要性证毕。

回到原题，现在只需要考虑选取  $G$  使得其满足尽可能多的  $G \bmod (a_{l_i} + a_{r_i}) \geq a_{l_i}$  即可。

考虑维护  $0 \leq G \leq m$  的答案数组。对  $a_{l_i} + a_{r_i} \geq \sqrt{m}$  的询问，可以转化为不超过  $O(\sqrt{m})$  次区间加操作。

利用差分和前缀和可以  $O(\sqrt{m})$  处理每个上面询问。

对  $a_{l_i} + a_{r_i}$  的询问，考虑用  $O(\sqrt{m})$  个长度不超过  $O(\sqrt{m})$  的数组  $c$  维护贡献。

对每个上面询问使得  $c(l_i+r_i)(l_i \sim r_i)$  加一。

最后从  $\sum m$  扫描一遍答案数组，同时加上这  $O(\sqrt{m})$  的数组的当前位置贡献，然后每个数组指针移动一位。

总时间复杂度  $O((m+q)\sqrt{m})$

```

const int MAXN=2e5+5,MAXM=500;
int a[MAXN],s[MAXN],c[MAXM][MAXM],p[MAXM];
int main()
{
    int T=read_int();
    while(T--){
        int n=read_int(),q=read_int(),m=read_int(),blk=sqrt(m)+1;
        _rep(i,0,m)s[i]=0;
        _for(i,1,blk){
            _for(j,0,i)c[i][j]=0;
            p[i]=0;
        }
        _rep(i,1,n)a[i]=read_int();
        while(q--){
            int l=read_int(),r=read_int();
            if(a[l]+a[r]>=blk){
                int pos=a[l];
                while(pos<=m){
                    s[pos]++;
                    if(pos+a[r]<=m)s[pos+a[r]]--;
                    pos+=a[l]+a[r];
                }
            }
            else{
                _for(i,l,l+r)
                c[l+r][i]++;
            }
        }
        _rep(i,1,m)s[i]+=s[i-1];
        _rep(i,0,m){
            _for(j,1,blk){
                s[i]+=c[j][p[j]];
                p[j]=(p[j]+1)%j;
            }
        }
        int ans=0;
        _rep(i,0,m)ans=max(ans,s[i]);
        enter(ans);
    }
    return 0;
}

```

}

## XOR Sums

### 题意

给定正整数序列  $A_1, A_2 \dots A_n$  接下来  $Q$  个询问。

定义一个集合的权值为该集合所有元素的异或和，每次询问元素个数不超过  $M$  的所有子集的权值和。

### 题解 1

考虑预处理出元素个数分别为  $\sim n$  的子集的权值和，然后  $O(1)$  处理询问。

按位处理贡献，假设第  $i$  位为  $1$  的数有  $c_1$  个，则第  $i$  位为  $0$  的数有  $c_0 = n - c_1$  个。

于是对子集元素个数为  $k$  的答案产生的贡献为

$$\sum_{j \in \text{odd}} \sum_{i \leq j \leq c_1} \binom{c_1}{j} \binom{c_0}{k-j}$$

令  $f(x) = \sum_{i \in \text{odd}} \sum_{i \leq i \leq c_1} \binom{c_1}{i} x^i$ ,  $g(x) = \sum_{i=0}^{c_0} \binom{c_0}{i} x^i$   
于是贡献为

$$\sum_{k=0}^{c_0} [x^k] f(x) g(x)$$

考虑  $\text{NTT}$  卷积，总时间复杂度  $O(n \log n \log v)$

ps. 比赛时考虑  $f_1(x) = (x+1)^{c_1}, f_2(x) = (-x+1)^{c_1}, g(x) = (x+1)^{c_0}$  然后贡献为  $\sum_{k=0}^{c_0} [x^k] \frac{(f_1(x) - f_2(x))g(x)}{2}$

然后脑抽使用了多项式  $\text{Pow}$  算  $f_1, f_2, g$  时间复杂度也是  $O(n \log n \log v)$  但直接  $T$  飞了，其实直接  $O(n)$  二项式就好了。

```
const int MAXN=2e5+5,Mod=998244353,inv2=(Mod+1)/2;
int quick_pow(int a,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*a%Mod;
        a=1LL*a*a%Mod;
        k>>=1;
    }
    return ans;
}
namespace Poly{
    const int G=3;
    int rev[MAXN<<2],Pool[MAXN<<3],*Wn[30];
    void init(){
```

```

int lg2=0,*pos=Pool,n,w;
while((1<<lg2)<MAXN*2)lg2++;
n=1<<lg2,w=quick_pow(G,(Mod-1)/(1<<lg2));
while(~lg2){
    Wn[lg2]=pos,pos+=n;
    Wn[lg2][0]=1;
    _for(i,1,n)Wn[lg2][i]=1LL*Wn[lg2][i-1]*w%Mod;
    w=1LL*w*w%Mod;
    lg2--;n>>=1;
}
int build(int k){
    int n, pos=0;
    while((1<<pos)<=k)pos++;
    n=1<<pos;
    _for(i,0,n)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
    return n;
}
void NTT(int *f,int n,bool type){
    _for(i,0,n)if(i<rev[i])
        swap(f[i],f[rev[i]]);
    int t1,t2;
    for(int i=1,lg2=1;i<n;i<=>1,lg2++){
        for(int j=0;j<n;j+=(i<<1)){
            _for(k,j,j+i){
                t1=f[k],t2=1LL*Wn[lg2][k-j]*f[k+i]%Mod;
                f[k]=(t1+t2)%Mod,f[k+i]=(t1-t2)%Mod;
            }
        }
        if(!type){
            reverse(f+1,f+n);
            int div=quick_pow(n,Mod-2);
            _for(i,0,n)f[i]=(1LL*f[i]*div%Mod+Mod)%Mod;
        }
    }
    void Mul(int *f,int _n,int *g,int _m,int xmod=0){
        int n=build(_n+_m-2);
        _for(i,_n,n)f[i]=0;_for(i,_m,n)g[i]=0;
        NTT(f,n,true);NTT(g,n,true);
        _for(i,0,n)f[i]=1LL*f[i]*g[i]%Mod;
        NTT(f,n,false);
        if(xmod)_for(i,xmod,n)f[i]=0;
    }
}
const int MAXS=30;
int c0[MAXS],c1[MAXS],ans[MAXN];
int frac[MAXN],inv[MAXN],f[MAXN<<2],g[MAXN<<2];
int C(int n,int m){
    return 1LL*frac[n]*inv[m]%Mod*inv[n-m]%Mod;
}

```

```
int main()
{
    Poly::init();
    int n=read_int();
    _for(i,0,n){
        int a=read_int();
        _for(j,0,MAXS){
            if(a&1)c1[j]++;
            else c0[j]++;
            a>>=1;
        }
    }
    frac[0]=1;
    _rep(i,1,n)frac[i]=1LL*i*frac[i-1]%Mod;
    inv[n]=quick_pow(frac[n],Mod-2);
    for(int i=n;i;i--)inv[i-1]=1LL*inv[i]*i%Mod;
    for(int i=MAXS-1;i>=0;i--){
        _rep(j,0,c1[i]){
            if(j%2==0)f[j]=0;
            else f[j]=C(c1[i],j);
        }
        _rep(j,0,c0[i])g[j]=C(c0[i],j);
        Poly::Mul(f,c1[i]+1,g,c0[i]+1);
        _rep(j,1,n)ans[j]=(2LL*ans[j]+f[j])%Mod;
    }
    _rep(i,1,n)ans[i]=(ans[i]+ans[i-1])%Mod;
    int q=read_int();
    while(q--)
        enter(ans[read_int()]);
    return 0;
}
```

## 题解 2

设  $\text{dp}(i,j,k)$  表示子集大小为  $i$  且考虑位数  $j$  且异或和为  $k$  的子集个数。

可以  $O(n \log v)$  得到答案，具体见该[代码](#)

## Another Tree with Number Theory

### 题意

给定一棵以  $1$  为根的树，接下来  $q$  次询问。

每次询问给定一个结点  $u$  初始权值  $w$  当结点  $u$  拥有权值  $w$  时，如果结点  $u$  是叶子结点，则该权值被吸收。

否则记 \$u\$ 的儿子结点数为 \$k\$。若 \$k \leq w\$ 则结点 \$u\$ 将权值均分为 \$k\$ 份并转移到每个儿子结点并继续下传，否则该权值损失。

问每次损失的权值。

## 题解

首先如果某个结点只有一个儿子，则将该结点与其儿子合并。

对结点 \$u\$ 设他到某个叶子结点的路径上所有点的儿子结点数的乘积为 \$k\$。若 \$k \leq w\$ 则该叶子结点能吸收权值 \$\frac{w}{k}\$；否则无法吸收。

考虑对每个结点用 \$\text{map}\$ 维护他每个 \$k\$ 对应的叶子节点数。

由于每次向上一级 \$k\$ 至少乘以 \$2\$，于是一个叶子结点最多对 \$O(\log w)\$ 个祖先节点有贡献。故维护 \$\text{map}\$ 的时间复杂度为 \$O(\log w \log n)\$。

对每个询问，用其存储在权值对应的桶中，然后跑一遍埃氏筛枚举因子计算贡献。

总时间复杂度 \$O(n \log n \log w + w \log w + q \times \text{divisors}(w) \log n)\$。

如果用哈希表替代 \$\text{map}\$ 时间复杂度为 \$(n \log w + w \log w + q \times \text{divisors}(w))\$。

ps. \$\text{map}\$ 访问空键默认返回 \$0\$ 但非常慢，不建议利用该特性，推荐用 \$\text{find}\$ 特判后再访问。

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:jxm2001:contest:cf\\_feb21&rev=1613475626](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:contest:cf_feb21&rev=1613475626)

Last update: 2021/02/16 19:40

