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 $p_1,p_2\cdots$ 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 $D\equiv A\mod B$□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 $A_1,A_2\cdots A_n$□保证 $A_i+A_1\ge A_{i+1}$□一开始由我方选定一个 G□使得 $0\le G\le M$□

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 $\{A_l,A_{l+1}\cdots A_r\}$ 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 $G\bmod (l+r)\ge l$。

接下来给出两个条件：

1. 每次可以拿 $S=\{a_1,a_2\cdots a_k\}$ 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 $a_i+\min S\lt a_j$，则存在 $a_k\in S$，使得 $a_i\lt a_k\lt a_j$。

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

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

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

如果选取 $a_{i+1}\sim a_k$，则 $G'\bmod (a_1+a_k)\gt 2a_1+a_i$，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 $0\le G\lt a_1+a_k$，易知 $0\lt G\lt a_1$ 是必败状态。

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

于是 $a_1\le G\lt a_k$ 是必胜状态。当 $a_k\le G\lt a_1+a_k$ 时取 a_k 个石头有 $G'\lt a_i$，于是 G 也是必胜状态。

于是 $0\le G\lt a_1+a_k$ 时必胜状态为 $a_1\le G\lt a_1+a_k$。

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

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

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

当 $(k+1)(a_1+a_k)+a_1\le G\lt (k+2)(a_1+a_k)$ 类比 $a_1\le G\lt a_1+a_k$ 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 $G\bmod (a_{l_i}+a_{r_i})\ge a_{l_i}$ 即可。

考虑维护 $0\le G\le m$ 的答案数组。对 $a_{l_i}+a_{r_i}\ge \sqrt m$ 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 $a_{l_i}+a_{r_i}\lt \sqrt m$ 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 $0\sim m$ 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;
```

```
}
```