Codeforces Round #654 (Div. 2)

比赛链接

E2. Asterism (Hard Version)

题意

给定 n 个正整数，代表 n 个敌人，每个敌人有 a_i 颗糖，再给定一个素数 $p(p \le n)$。

定义游戏规则为一开始玩家拥有若干颗糖，每次玩家可以选取一个未挑战且手上糖果数不大于玩家的敌人，打败他获得一颗糖果。

定义 $f(x)$ 为玩家一开始拥有 x 颗糖时可以战胜所有敌人的方案数。

要求输出所有满足 $p\nmid f(x)$ 的 x。

题解 1

设 b_i 为糖果数不大于 i 的敌人个数。$C_i(x)=b_{x+i}-i$。

有 $f(x)=\prod_{i=0}^{n-1} C_i(x)$。

易知 $C_{n-1}(x)\le 1$ 且 $C_i(x)-C_{i-1}(x)=b_{x+i}-1\ge -1$。

所以 $p\nmid f(x)\iff \forall i(0\le i\lt n \longrightarrow C_i\lt p)$。

由于 $C_i(x)\le C_i(x+1)$。所以答案为一段区间。

考虑 x 的可能范围，记 $A=\max a_i$。若 $x\le A-n$。则 $f(x)= 0$。有 $p\mid f(x)$。若 $x\ge A$。则 $f(x)= n!$。有 $p\mid f(x)$。

所以将区间左端点初值设为 $A-n+1$。右端点初值设为 A。暴力枚举 $i=0\sim n-1$ 的情况，维护一下即可，时间复杂度 $O(n)$。

```
const int MAXN=1e5+5;
int a[MAXN],b[MAXN<<1];
int main()
{
    int n=read_int(),p=read_int(),A=0;
    _for(i,0,n){
        a[i]=read_int();
        A=max(A,a[i]);
    }
    _for(i,0,n)
    b[max(0,a[i]-(A-n))]++;
    _for(i,1,n<<1)
```

```
    b[i]+=b[i-1];
    int st=1,en=n;
    _rep(i,1,n){
        while(b[st+(i-1)]<i)
        st++;
    }
    _for(i,0,n){
        while(b[en+i]-i>=p)
        en--;
    }
    st+=(A-n),en+=(A-n);
    if(st<=en){
        enter(en-st+1);
        _rep(i,st,en){
            if(i!=st)
            putchar(' ');
            write(i);
        }
    }
    else
    enter(0);
    return 0;
}
```

题解 2

$f(x)=\prod_{i=0}^{n-1} C_i(x)=\prod_{i=0}^{n-1} b_{x+i}-i=\prod_{i=x}^{x+n-1} b_{i}-i+x$

所以 $p\nmid f(x)\iff$ 对所有 $x\le i\lt x+n$有 x 与 $i-b_{i}$ 不同余 $\iff$ $x-(A-n)$ 与 $i-(A-n)-b_{i}$ 不同余。

暴力枚举 $A-n\lt x\lt A$考虑 $i-b_{i}$ 对于 x 与 $x-1$ 的约束只用一项不同，所以可以用滑动窗口维护。

```
const int MAXN=1e5+5;
int a[MAXN],b[MAXN<<1],c[MAXN],ans[MAXN],cnt;
int mod(int a,int p){return (a%p+p)%p;}
int main()
{
    int n=read_int(),p=read_int(),A=0;
    _for(i,0,n){
        a[i]=read_int();
        A=max(A,a[i]);
    }
    _for(i,0,n)
    b[max(0,a[i]-(A-n))]++;
    _for(i,1,n<<1)
    b[i]+=b[i-1];
```

```
    _for(i,0,n)
    c[mod(i-b[i],p)]++;
    _for(i,1,n){
        c[mod(i-1-b[i-1],p)]--;
        c[mod(i+n-1-b[i+n-1],p)]++;
        if(!c[i%p])
        ans[cnt++]=i+A-n;
    }
    enter(cnt);
    _for(i,0,cnt){
        if(i)
        putchar(' ');
        write(ans[i]);
    }
    return 0;
}
```

F. Raging Thunder

题意

~~题目太长了，所以省略了~~

题解

序列操作，很明显用线段树维护。

每个区间考虑维护该区间的前缀和后缀以及非前后缀的最大答案。

同时每个区间还需要维护一个 type 参数来区间是有前后缀的区间还是不分前后缀的区间。

关于区间合并，考虑将 '< >' 可作为串的划分标志，'<' 左边的串和 '>' 右边的串互不关联。

另外每个线段树结点需要同时维护两个区间，一个为正常区间，一个为 '<' 和 '>' 互换的区间，便于修改和查询操作。

还有关于 push_up 操作要注意提前下放子节点的懒标记，否则会出错。

时间复杂度 $O(n+q\log n)$，细节见代码。

```
const int MAXN=5e5+5;
struct Node{
    int str[2][2],ans;
    bool type;
}node[MAXN<<2][2];
int lef[MAXN<<2],rig[MAXN<<2];
bool isleaf[MAXN<<2],lazy[MAXN<<2];
char buf[MAXN];
```

```
Node merge(const Node &lef,const Node &rig){
    Node temp;
    temp.ans=max(lef.ans,rig.ans);
    if(lef.type&&rig.type){
        temp.type=true;
        temp.str[0][0]=lef.str[0][0];
        temp.str[0][1]=lef.str[0][1];
        temp.str[1][0]=rig.str[1][0];
        temp.str[1][1]=rig.str[1][1];
        if(lef.str[1][1]&&rig.str[0][0])
temp.ans=max(max(lef.str[1][0]+lef.str[1][1],rig.str[0][0]+rig.str[0][1]),t
emp.ans);
        else
temp.ans=max(lef.str[1][0]+lef.str[1][1]+rig.str[0][0]+rig.str[0][1],temp.a
ns);
    }
    else if(lef.type){
        temp.type=true;
        temp.str[0][0]=lef.str[0][0];
        temp.str[0][1]=lef.str[0][1];
        if(lef.str[1][1]&&rig.str[0][0]){
            temp.ans=max(lef.str[1][0]+lef.str[1][1],temp.ans);
            temp.str[1][0]=rig.str[0][0];
            temp.str[1][1]=rig.str[0][1];
        }
        else{
            temp.str[1][0]=lef.str[1][0]+rig.str[0][0];
            temp.str[1][1]=lef.str[1][1]+rig.str[0][1];
        }
    }
    else if(rig.type){
        temp.type=true;
        temp.str[1][0]=rig.str[1][0];
        temp.str[1][1]=rig.str[1][1];
        if(lef.str[0][1]&&rig.str[0][0]){
            temp.ans=max(rig.str[0][0]+rig.str[0][1],temp.ans);
            temp.str[0][0]=lef.str[0][0];
            temp.str[0][1]=lef.str[0][1];
        }
        else{
            temp.str[0][0]=lef.str[0][0]+rig.str[0][0];
            temp.str[0][1]=lef.str[0][1]+rig.str[0][1];
        }
    }
    else{
        if(lef.str[0][1]&&rig.str[0][0]){
            temp.type=true;
            temp.str[0][0]=lef.str[0][0];
            temp.str[0][1]=lef.str[0][1];
            temp.str[1][0]=rig.str[0][0];
```

```
            temp.str[1][1]=rig.str[0][1];
        }
        else{
            temp.type=false;
            temp.str[0][0]=lef.str[0][0]+rig.str[0][0];
            temp.str[0][1]=lef.str[0][1]+rig.str[0][1];
            temp.str[1][0]=temp.str[1][1]=0;
        }
    }
    return temp;
}
void push_down(int k){
    if(lazy[k]){
        swap(node[k][0],node[k][1]);
        lazy[k]=false;
        if(!isleaf[k])
        lazy[k<<1]^=1,lazy[k<<1|1]^=1;
    }
}
void push_up(int k){
    push_down(k<<1);push_down(k<<1|1);
    node[k][0]=merge(node[k<<1][0],node[k<<1|1][0]);
    node[k][1]=merge(node[k<<1][1],node[k<<1|1][1]);
}
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R;
    int M=L+R>>1;
    if(L==R){
        isleaf[k]=true;
        if(buf[M]=='<'){
            node[k][0].str[0][1]=1;
            node[k][1].str[0][0]=1;
        }
        else{
            node[k][0].str[0][0]=1;
            node[k][1].str[0][1]=1;
        }
        return;
    }
    build(k<<1,L,M);
    build(k<<1|1,M+1,R);
    push_up(k);
}
void update(int k,int L,int R){
    if(L<=lef[k]&&rig[k]<=R)
    return lazy[k]^=1,void();
    push_down(k);
    int mid=lef[k]+rig[k]>>1;
    if(mid<L)
    update(k<<1|1,L,R);
    else if(mid>=R)
```

```
    update(k<<1,L,R);
    else{
        update(k<<1,L,R);
        update(k<<1|1,L,R);
    }
    push_up(k);
}
Node query(int k,int L,int R){
    push_down(k);
    if(L<=lef[k]&&rig[k]<=R)
    return node[k][0];
    int mid=lef[k]+rig[k]>>1;
    if(mid<L)
    return query(k<<1|1,L,R);
    else if(mid>=R)
    return query(k<<1,L,R);
    else
    return merge(query(k<<1,L,R),query(k<<1|1,L,R));
}
int main()
{
    int n=read_int(),q=read_int(),l,r;
    Node temp;
    scanf("%s",buf+1);
    build(1,1,n);
    while(q--){
        l=read_int(),r=read_int();
        update(1,l,r);
        temp=query(1,l,r);
enter(max(temp.ans,max(temp.str[0][0]+temp.str[0][1],temp.str[1][0]+temp.st
r[1][1])));
    }
    return 0;
}
```