

[比赛链接](#)

## 补题情况

题目	蒋贤蒙	王赵安	王智彪
A	0	0	0
B	2	2	0
C	0	0	0
D	0	0	0
E	1	0	2
G	0	0	0
J	2	2	0
K	2	0	0

## 题解

### B. xay loves monotonicity

#### 题意

给定一个序列  $A$  和序列  $B$  其中  $0 \leq b_i \leq 1$  接下来三种操作：

1.  $a_i \leftarrow t$
2. 对  $i \in [r]$   $b_i \leftarrow b_i + 1$
3. 给定  $l, r$  选取最长下标序列  $i_1 \leq i_2 \leq \dots \leq i_k \in [r]$  满足  $a_{i_1} \leq a_{i_2} \leq \dots \leq a_{i_k}$  且对任意  $i_t < i_{t+1}$  有  $a_{i_t} < a_{i_{t+1}}$

对每个操作  $s$ ，输出  $b_{i_t} \neq b_{i_{t+1}}$  的个数。

#### 题解

设  $ma(L, R) = \max(a[L \sim R])$ ,  $mb(L, R)$  表示  $ma(L, R)$  对应的  $b_i$  如果存在多个就取最右边的。

设  $\text{query}(L, R, p, q)$  表示假如当前序列末尾对应  $a_i = p, b_i = q$  时遍历区间  $(L, R)$  得到的答案。

于是，如果  $a_i > ma(L, M)$  则  $\text{query}(L, R, p, q) = \text{query}(M+1, R, p, q)$

否则，有  $\text{query}(L, R, p, q) = \text{query}(L, M, p, q) + \text{query}(M+1, R, ma(L, M), mb(L, M))$

建立线段树，每个区间维护  $\text{query}(M+1, R, ma(L, M), mb(L, M))$

这样，对一个询问，如果该询问正好对应一个线段树区间，则查询复杂度  $O(\log n)$

否则，将该询问拆分成  $O(\log n)$  个线段树区间，串联查询计算答案，时间复杂度  $O(\log^2 n)$

对与修改操作，修改完暴力询问更新  $\text{query}(M+1, R, ma(L, M), mb(L, M))$  由于这是线段树区间，

所以复杂度为  $O(\log n)$

所以修改的总复杂度也是  $O(\log^2 n)$  总时间复杂度  $O(n \log n + q \log^2 n)$

ps. 比赛写了  $O(nq)$  的假算法，居然过了。

```
const int MAXN=2e5+5;
int a[MAXN],b[MAXN];
int lef[MAXN<<2],rig[MAXN<<2],s[MAXN<<2],tag[MAXN<<2];
struct Node{
    int a,b;
}mv[MAXN<<2];
Node Max(Node L,Node R){
    if(L.a>R.a)
        return L;
    else
        return R;
}
void push_tag(int k){
    mv[k].b^=1;
    tag[k]^=1;
}
void push_down(int k){
    if(tag[k]){
        push_tag(k<<1);
        push_tag(k<<1|1);
        tag[k]=0;
    }
}
int query(int k,int a,int b){
    if(lef[k]==rig[k])
        return mv[k].a>=a&&mv[k].b!=b;
    push_down(k);
    if(a<=mv[k<<1].a)
        return query(k<<1,a,b)+s[k];
    else
        return query(k<<1|1,a,b);
}
void push_up(int k){
    mv[k]=Max(mv[k<<1],mv[k<<1|1]);
    s[k]=query(k<<1|1,mv[k<<1].a,mv[k<<1].b);
}
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R;
    int M=L+R>>1;
    if(L==R){
        mv[k]=Node{a[M],b[M]};
        return;
    }
    build(k<<1,L,M);
```

```

    build(k<<1|1,M+1,R);
    push_up(k);
}
Node query_max(int k,int L,int R){
    if(L<=lef[k]&&rig[k]<=R)
        return mv[k];
    push_down(k);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=R)
        return query_max(k<<1,L,R);
    else if(mid<L)
        return query_max(k<<1|1,L,R);
    else
        return Max(query_max(k<<1,L,R),query_max(k<<1|1,L,R));
}
int query(int k,int L,int R,int a,int b){
    if(L<=lef[k]&&rig[k]<=R)
        return query(k,a,b);
    push_down(k);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=R)
        return query(k<<1,L,R,a,b);
    else if(mid<L)
        return query(k<<1|1,L,R,a,b);
    else{
        Node t=query_max(k<<1,L,R);
        if(a<=t.a)
            return query(k<<1,L,R,a,b)+query(k<<1|1,L,R,t.a,t.b);
        else
            return query(k<<1|1,L,R,a,b);
    }
}
void update1(int k,int pos,int v){
    if(lef[k]==rig[k]){
        mv[k].a=v;
        return;
    }
    push_down(k);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=pos)
        update1(k<<1,pos,v);
    else
        update1(k<<1|1,pos,v);
    push_up(k);
}
void update2(int k,int L,int R){
    if(L<=lef[k]&&rig[k]<=R){
        push_tag(k);
        return;
    }
    push_down(k);
}

```

```
int mid=lef[k]+rig[k]>>1;
if(mid>=L)
update2(k<<1,L,R);
if(mid<R)
update2(k<<1|1,L,R);
push_up(k);
}
int main(){
int n=read_int();
_rep(i,1,n)a[i]=read_int();
_rep(i,1,n)b[i]=read_int();
build(1,1,n);
int q=read_int();
while(q--){
int opt=read_int(),t1=read_int(),t2=read_int();
if(opt==1)
update1(1,t1,t2);
else if(opt==2)
update2(1,t1,t2);
else{
if(t1==t2)
enter(0);
else{
Node t=query_max(1,t1,t1);
enter(query(1,t1+1,t2,t.a,t.b));
}
}
}
return 0;
}
```

## E. xay loves nim

### 题意

给了  $n$  堆石子和  $m$  个可以取走的石子的数量，记为  $x_i$ 。除了这  $m$  种石子，还可以取莫比乌斯函数值为  $1$  的数量石子。同时给出这  $n$  堆石子的数量范围  $[l_i, r_i]$ 。求所有的情况中，先手必胜的局数，局面不同当且仅当存在一堆石子在两个局面中数量不同。

$$1 \leq n \leq 10^6, 1 \leq l_i, r_i \leq 10^5, 1 \leq m \leq 5, 1 \leq x_i \leq 10^5$$

### 题解

大综合题，显然需要会莫比乌斯反演（废话），还需要会博弈论的  $sg$  那套理论，先手必胜当且仅当所有的  $sg$  值异或起来不为  $0$ 。听出题人说  $sg$  值最高不会超过  $230$ ，但是现场的时候怎么知道嘛...

显然需要先筛出来  $1$  到  $100000$  的莫比乌斯函数值，才能知道哪些能加进去，然后再把  $m$  种数字也

放进去，然后就是看每个状态的后继状态，这里也是学到了可以暴力搞。

对于每一个新的值  $i$  搞一个数组，看这个值的后继状态有没有  $sg$  为这个值的，如果有需要继续找，直到找不到，根据  $mex$  那套理论，就是此时的  $sg$  了。然后  $i$  加上刚才放进去的那些数的后继状态可以到达  $i$  于是这些状态的后继  $sg$  值可以有现在这个值，这里我们只关心能不能有，所以可以用  $bitset$  优化一下，这部分的复杂度是  $O(\frac{100000^2}{w} + 256 \times 100000)$  这里本来要写  $230$  的，但是后面需要变成  $256$ 。

于是我们处理出了所有石子数的  $sg$  值，接下来对于每一堆石子，我们可以求出来，他们这个范围内，分别有多少个  $sg$  值为  $0$  的，有多少个  $sg$  值为  $1$  的，依次类推。

然后我们需要关心有多少个  $a[1] \wedge a[2] \wedge \dots \wedge a[n] \neq 0$  这玩意显然可以看成  $FWT$  我们先统计出每个区间内有多少个  $sg$  值为  $i$  的，这显然前缀和就可以。然后我们相当于看  $n$  个多项式相乘，每个多项式的幂次是各个  $sg$  值，系数是有多少个状态的  $sg$  值是这个值。如果对于每个求  $FWT$  再乘起来，最后再  $IFWT$  我们需要开到  $256$ ，这就照应了前面。算了一下，单个复杂度是  $O(256 \times \log_2(256)) = O(2048)$  然后  $n$  个就是  $2 \times 10^9$  的，这显然是自杀行为。

然后  $oi-wiki$  上一段话刷新了我对  $FWT$  的认知：若我们令  $i \& j$  中  $1$  的奇偶性为  $i$  与  $j$  的奇偶性，于是  $i$  与  $k$  的奇偶性异或  $j$  与  $k$  的奇偶性等于  $i \wedge j$  与  $k$  的奇偶性。然后可以得到异或的  $FWT$   $A_i = \sum_{C_1} A_j - \sum_{C_2} A_j$   $C_1$  表示  $i \& j$  的奇偶性为偶  $C_2$  表示  $i \& j$  的奇偶性为奇。（其实记住就行...）

又因为  $sg$  的范围有限，所以我们可以先预处理出每个数的二进制有多少位为  $1$ （或者直接  $\_builtin\_popcount$  也可以）。

然后对于前缀和数组，就不能直接暴力加一了，判断两者与的奇偶性，如果为偶，则加一，不然减一。然后这样就直接把每一堆的  $FWT$  数组给求完了，复杂度变成  $O(256n)$  而不是原来的  $O(2048n)$  再全乘起来求一个  $IFWT$  就可以了，整个这部分的复杂度都是  $O(256n)$  的，最后对于所有  $sg$  值不为  $0$  的结果都加起来就是答案了。

另外这题卡常！前缀和数组必须大的做第一维，我不倒过来会  $st$  掉绝大多数数据，倒过来效率就第一了...

```
const int maxn=100000,N=1e6+10,maxm=256,MOD=1e9+7,inv2=500000004;
bool check[maxn+1];
int prime[maxn+1],mu[maxn+1];
int
tot,n,m,ls[N],rs[N],sg[maxn+1],sum[maxn+1][maxm+1],ans[maxm+1],o[maxm+1];
bitset<maxn+1> t,b[maxm];

void Moblus() {
    mu[1]=1;
    for(int i=2; i<=maxn; i++) {
        if(!check[i]) {
            mu[i]=-1;
            prime[tot++]=i;
        }
        for(int j=0; j<tot; j++) {
            if(i*prime[j]>maxn) break;
            check[i*prime[j]]=true;
            if(i%prime[j]==0) {
```

```
        mu[i*prime[j]]=0;
        break;
    } else {
        mu[i*prime[j]]=-mu[i];
    }
}
}
}

void xor_FWT(int *P,int opt,int N) {
    for(int i=2; i<=N; i<=1)
        for(int p=i>>1,j=0; j<N; j+=i)
            for(int k=j; k<j+p; ++k) {
                int x=P[k],y=P[k+p];
                P[k]=((ll)x+y)%MOD;
                P[k+p]=((ll)x-y+MOD)%MOD;
            }
    if(opt==-1)P[k]=(ll)P[k]*inv2%MOD,P[k+p]=(ll)P[k+p]*inv2%MOD;
}

void init() {
    o[0]=0;
    for(int i=1;i<maxm;i++)o[i]=o[i>>1]+(i&1);
    for(int i=1; i<=maxn; i++) {
        if(mu[i]==1)t.set(i);
    }
    for(int i=0; i<=maxn; i++) {
        sg[i]=0;
        while(b[sg[i]][i]) sg[i]++;
        b[sg[i]]|=(t<<i);
    }
}

int main() {
    Moblus();
    read(n);read(m);
    for(int i=1; i<=n; i++) read(ls[i]),read(rs[i]);
    for(int i=1,tmp; i<=m; i++) read(tmp),t.set(tmp);
    init();
    for(int i=0; i<=maxn; i++)
        for(int j=0; j<maxm; j++)
            if(!(o[sg[i]&j]&1))sum[i][j]=1;
            else sum[i][j]=MOD-1;
    for(int i=1; i<=maxn; i++) {
        for(int j=0; j<maxm; j++) {
            sum[i][j]=sum[i][j]+sum[i-1][j];
            if(sum[i][j]>=MOD) sum[i][j]-=MOD;
        }
    }
    for(int i=0; i<maxm; i++)ans[i]=1;
}
```

```

for(int i=1; i<=n; i++)
    for(int j=0; j<maxm; j++)
        ans[j]=(ll)ans[j]*(sum[rs[i]][j]-sum[ls[i]-1][j])%MOD;
xor_FWT(ans, -1, maxm);
ll sum=0;
for(int i=1; i<maxm; i++) {
    sum=sum+ans[i];
    if(sum>=MOD) sum-=MOD;
}
printf("%lld\n", (sum+MOD)%MOD);
return 0;
}

```

## J. xay loves Floyd

### 题意

给定一个有向图，初始时  $\text{dis}(u,u)=0, \text{dis}(u,v)=\infty (u \neq v)$

接下来给定若干条边  $(u,v,w)$  使得  $\text{dis}(u,v)=w$  询问以下两个程序最终结果中满足  $\text{dis}(u,v)$  相同的  $(u,v)$  对数。

```

for k from 1 to n
  for i from 1 to n
    for j from 1 to n
      dis[i][j] <- min(dis[i][j], dis[i][k] + dis[k][j])

```

```

for i from 1 to n
  for j from 1 to n
    for k from 1 to n
      dis[i][j] <- min(dis[i][j], dis[i][k] + dis[k][j])

```

### 题解

首先  $n$  次单点源最短路算法  $O(nm \log m)$  求出  $\text{dis}$  的真实值。

设  $ok(u,v)$  表示  $\text{dis}(u,v)$  是否为正确值，考虑第二个程序得到的  $\text{dis}(i,j)$  正确的充要条件。

不难发现，只要  $i$  到  $j$  的最短路上有一点  $k$  满足  $ok(i,k) \wedge ok(k,j)$  即可。

首先考虑找到所有满足条件的  $k$  设  $\text{path}(u,v)$  表示  $u$  到  $v$  上最短路上的点集，于是有状态转移方程

$$\text{path}(i,j) = \bigcup_{\text{dis}(i,k) + w(k,j) = \text{dis}(i,j)} \text{path}(i,k)$$

对固定的  $i$  考虑将  $j$  按  $\text{dis}(i,j)$  从小到大排序后用  $\text{bitset}$  加速上述转移。

然后按  $1 \sim n$  顺序枚举  $j$  于是有  $ok(i,j) = \sum_{k=1}^n ok(i,k) \wedge ok(k,j) \wedge (k \in$

$\text{path}(i,j)$

用两种  $\text{bitset}$  维护  $\text{ok}(i,\text{last}),\text{ok}(\text{last},i)$  上述转移也可以用  $\text{bitset}$  加速。总时间复杂度  $O(nm\log m + \frac{n^2m}{w})$

```
const int MAXN=2e3+5,MAXM=5e3+5,inf=1e9;
struct Edge{
    int to,w,next;
}edge[MAXM];
int head[MAXN],edge_cnt;
void Insert(int u,int v,int w){
    edge[++edge_cnt]=Edge{v,w,head[u]};
    head[u]=edge_cnt;
}
namespace DJ{
    bool vis[MAXN];
    void solve(int n,int s,int *dis){
        _rep(i,1,n){
            dis[i]=inf;
            vis[i]=false;
        }
        dis[s]=0;
        priority_queue<pair<int,int> > q;
        q.push(make_pair(0,s));
        while(!q.empty()){
            int u=q.top().second;q.pop();
            if(vis[u])continue;
            vis[u]=true;
            for(int i=head[u];i;i=edge[i].next){
                int v=edge[i].to;
                if(dis[v]>dis[u]+edge[i].w){
                    dis[v]=dis[u]+edge[i].w;
                    q.push(make_pair(-dis[v],v));
                }
            }
        }
    }
}
int dis[MAXN][MAXN],d0[MAXN][MAXN];
bitset<MAXN> ok1[MAXN],ok2[MAXN],path[MAXN];
int main(){
    int n=read_int(),m=read_int();
    _rep(i,1,n)_rep(j,1,n)d0[i][j]=inf;
    _rep(i,1,n)d0[i][i]=0;
    while(m--){
        int u=read_int(),v=read_int(),w=read_int();
        d0[u][v]=w;
        Insert(u,v,w);
    }
    _rep(i,1,n)
```

```

DJ::solve(n,i,dis[i]);
_rep(i,1,n)_rep(j,1,n){
    if(dis[i][j]==d0[i][j])
        ok1[i][j]=ok2[j][i]=true;
}
_rep(u,1,n){
    vector<pair<int,int> >vec;
    _rep(v,1,n){
        vec.push_back(make_pair(dis[u][v],v));
        path[v].reset();
    }
    sort(vec.begin(),vec.end());
    for(pair<int,int> p:vec){
        int v=p.second;
        path[v][v]=true;
        for(int i=head[v];i;i=edge[i].next){
            int t=edge[i].to;
            if(dis[u][v]+edge[i].w==dis[u][t])
                path[t]=path[v];
        }
    }
    _rep(v,1,n){
        if((ok1[u]&ok2[v]&path[v]).any())
            ok1[u][v]=ok2[v][u]=true;
    }
}
int ans=0;
_rep(i,1,n)
ans+=ok1[i].count();
enter(ans);
return 0;
}

```

## K. xay loves sequence

### 题意

给定一个长度为  $n$  的序列  $A$  接下来若干询问，每次输出  $f(l,r,k)$

定义  $f(l,r,k)$  表示将  $A$  的子串  $a[l \sim r]$  全部变为  $0$  的最小操作次数。

其中每次操作为选择  $a[l \sim r]$  的一个子串  $a[l_2 \sim r_2]$  令  $a_i \equiv a_{i+1} \pmod k (l_2 \leq i \leq r_2)$  或者  $a_i \equiv a_{i-1} \pmod k (l_2 \leq i \leq r_2)$

保证对所有  $k$  满足  $k > a_i$

### 题解

对每次询问的  $a[l \sim r]$  令  $a_{l-1}=0, a_{r+1}=0$  设  $b_i = a_i - a_{i-1}$  ( $l \leq i \leq r+1$ )

于是每次操作等价于选取一对  $(i, j)$   $b_i \equiv b_{i+1} \pmod k, b_j \equiv b_{j-1} \pmod k$

同时  $\sum_{i=l}^{r+1} b_i = a_{r+1} - a_{l-1} = 0$  最终目标是将  $b_i$  全部变为  $0$ 。在不考虑取模的情况下，最小费用显然为  $\frac{\sum_{i=l}^{r+1} |b_i|}{2}$

考虑取模，则最终有  $b_i = 0 \pmod k$  且仍然有  $\sum_{i=l}^{r+1} b_i = 0$

考虑将一些  $b_i > 0$  的数目标设为  $k$  则对操作数的影响为  $\frac{k-2b_i}{2}$  将一些  $b_i \leq 0$  的数目标设为  $-k$  则对操作数的影响为  $\frac{k+2b_i}{2}$

由于要保证  $\sum_{i=l}^{r+1} b_i = 0$  所以可以设最终有  $x$  个  $b_i = k$  同时有  $x$  个  $b_i = -k$

分别在  $b_i > 0$  和  $b_i \leq 0$  的两个组数中取原来绝对值前  $x$  大的  $b_i$  显然是最优的。另外随  $x$  增大收益显然具有单峰性。

于是二分答案即可。另外对于区间询问可以用主席树维护  $b[l+1 \sim r]$  的值，然后补充  $a_l, -a_r$

时间复杂度  $O(n \log n \log v)$  空间复杂度  $O(n \log v)$

```
const int MAXN=2e5+5,MAXV=(1<<30)+5;
struct Node{
    int lch,rch,cnt;
    LL sum;
};
Node node[MAXN*100];
int node_cnt,root1[MAXN],root2[MAXN];
int a[MAXN],b[MAXN];
LL c[MAXN];
int nodecopy(int k){
    node[++node_cnt]=node[k];
    return node_cnt;
}
#define rch(k) node[node[k].rch]
void update(int &k,int p,int pos,LL lef=0,LL rig=MAXV){
    k=nodecopy(p);
    node[k].cnt++;
    node[k].sum+=pos;
    if(lef==rig)
        return;
    LL mid=lef+rig>>1;
    if(mid>=pos)
        update(node[k].lch,node[p].lch,pos,lef,mid);
    else
        update(node[k].rch,node[p].rch,pos,mid+1,rig);
}
int query_val(int k1,int k2,int rk,LL lef=0,LL rig=MAXV){
    LL mid=lef+rig>>1;
    if(lef==rig)
        return mid;
```

```

    int cnt=rch(k2).cnt-rch(k1).cnt;
    if(rk>cnt)
        return query_val(node[k1].lch,node[k2].lch,rk-cnt,lef,mid);
    else
        return query_val(node[k1].rch,node[k2].rch,rk,mid+1,rig);
}
LL query_sum(int k1,int k2,int rk,LL lef=0,LL rig=MAXV){
    LL mid=lef+rig>>1;
    if(lef==rig)
        return 1LL*rk*mid;
    int cnt=rch(k2).cnt-rch(k1).cnt;
    if(rk>cnt)
        return rch(k2).sum-rch(k1).sum+query_sum(node[k1].lch,node[k2].lch,rk-
cnt,lef,mid);
    else
        return query_sum(node[k1].rch,node[k2].rch,rk,mid+1,rig);
}
int main(){
    int n=read_int(),q=read_int();
    _rep(i,1,n)a[i]=read_int();
    _rep(i,1,n){
        b[i]=a[i]-a[i-1];
        c[i]=c[i-1]+abs(b[i]);
        root1[i]=root1[i-1];
        root2[i]=root2[i-1];
        if(b[i]>=0)
            update(root1[i],root1[i],b[i]);
        else
            update(root2[i],root2[i],-b[i]);
    }
    while(q--){
        int l=read_int(),r=read_int(),k=read_int();
        int rt1=root1[r],p1=root1[l],rt2=root2[r],p2=root2[l];
        if(a[l]>=0)
            update(rt1,rt1,a[l]);
        else
            update(rt2,rt2,-a[l]);
        if(-a[r]>=0)
            update(rt1,rt1,-a[r]);
        else
            update(rt2,rt2,a[r]);
        int lef=1,rig=min(node[rt1].cnt-node[p1].cnt,node[rt2].cnt-
node[p2].cnt),rk=0;
        LL ans=c[r]-c[l]+a[l]+a[r];
        while(lef<=rig){
            int mid=lef+rig>>1;
            if(query_val(p1,rt1,mid)+query_val(p2,rt2,mid)>k){
                rk=mid;
                lef=mid+1;
            }
            else

```

```
        rig=mid-1;
    }
    if(rk!=0)
        ans-=(query_sum(p1,rt1,rk)+query_sum(p2,rt2,rk)-1LL*k*rk)*2;
        enter(ans/2);
    }
    return 0;
}
```

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

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:%E7%BB%84%E9%98%9F%E8%AE%AD%E7%BB%83%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95:contest12&rev=1628847316](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%BB%84%E9%98%9F%E8%AE%AD%E7%BB%83%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95:contest12&rev=1628847316)

Last update: 2021/08/13 17:35