令 $p_i=\max\limits_{j<i\wedge a_j=a_i} j$，那么 $\operatorname{mex}(\{a_l,\dots,a_r\})=\min\limits_{p_j<l\le r<j} a_j$（其实就是对每个数把序列分段）。

提取出所有 (p_j,j) 之后就是单点增删，矩形查权值 min。

这个树套树，K-D Tree，分块都可以做，能不能过另说。

讲一讲分块做法。

首先对序列（右端点 i）和值域（左端点 p_i）都分一次块。

因为每次查询的是右端点的后缀，所以预处理出 $suf_{i,j}$ 表示序列第 $i\sim\lceil n/B\rceil$ 块中值域在第 j 个块的权值 min。

修改第 j 个块中的数的时候就把对所有 i 重构 $suf_{i,j}$。

查询时整块直接查 $suf_{i,j}$，散块暴力判断是否满足条件。因为 i 和 p_i 是一一对应的，所以散块包含的点对个数是 $O(B)$ 的。

注意到序列开头的 i 不一定存在 p_i，而结尾的 p_i 不一定有与之对应的 i，从而难以在散块时查到合适的值。因此在序列首尾分别插入 $0\sim n-1$ 保证一定存在对于每个 i 均存在 j,k 满足 $p_i=j,p_k=i$。

时间复杂度 $O(n\sqrt{n})$，空间复杂度 $O(n)$。

```cpp
#include<bits/stdc++.h>

using namespace std;

ifstream fin("mex.in");
ofstream fout("mex.out");
#define cin fin
#define cout fout
#define endl '\n'

const int N=3e5+9;
const int S=6e2+9;

bool bg;
set<int> s[N];
int a[N],lp[N],rp[N],n,m,q,T;
int iblk[N],iL[S],iR[S],B;
int suf[S][S];
bool st;
inline void Init(){
    B=sqrt(m);
    for(int i=1;i<=m;i++) iblk[i]=(i-1)/B+1;
    for(int i=1;i<=m;i++) iR[iblk[i]]=i;
    for(int i=m;i>=1;i--) iL[iblk[i]]=i;
    for(int i=1;i<=iblk[m];i++){
        for(int j=1;j<=iblk[m];j++) suf[j][i]=n;
    }
}
inline void Rebuild(int x){
    for(int i=1;i<=iblk[m];i++) suf[x][i]=n;
    for(int i=iL[x];i<=iR[x];i++){
```

```
        if(rp[i]) suf[x][iblk[rp[i]]]=min(suf[x][iblk[rp[i]]],a[rp[i]]);
    }
    for(int i=iblk[m]-1;i>=1;i--) suf[x][i]=min(suf[x][i],suf[x][i+1]);
}
inline int Query(int l,int r){
    int ans=n;
    if(iblk[r]<iblk[m]){
        for(int i=1;i<iblk[l];i++) ans=min(ans,suf[i][iblk[r]+1]);
    }
    for(int i=iL[iblk[l]];i<=l;i++){
        if(rp[i]&&rp[i]>=r) ans=min(ans,a[rp[i]]);
    }
    for(int i=r;i<=iR[iblk[r]];i++){
        if(lp[i]&&lp[i]<=l) ans=min(ans,a[i]);
    }
    return ans;
}
inline void Insert(int x){
    int l=0,r=0;
    auto ir=s[a[x]].upper_bound(x);
    if(ir!=s[a[x]].end()) r=*ir;
    if(ir!=s[a[x]].begin()) l=*prev(ir);
    s[a[x]].insert(x);
    if(l) rp[l+1]=x,lp[x]=l+1,Rebuild(iblk[l+1]);
    else lp[x]=0;
    if(r) rp[x+1]=r,lp[r]=x+1,Rebuild(iblk[x+1]);
    else rp[x+1]=0;
}
inline void Erase(int x){
    int l=0,r=0;
    s[a[x]].erase(x);
    auto ir=s[a[x]].upper_bound(x);
    if(ir!=s[a[x]].end()) r=*ir;
    if(ir!=s[a[x]].begin()) l=*prev(ir);
    if(l) rp[l+1]=r,lp[x]=0,Rebuild(iblk[l+1]);
    if(r) rp[x+1]=0,lp[r]=l+1,Rebuild(iblk[x+1]);
}

signed main(){
    cin>>T>>n>>q;
    for(int i=1;i<=n;i++) cin>>a[i+n],a[i+n]=min(a[i+n],n-1);
    for(int i=1;i<=n;i++) a[i]=a[i+n+n]=i-1;
    m=3*n;

    Init();
    for(int i=1;i<=n;i++) Insert(i);
    for(int i=1;i<=n;i++) Insert(i+n);
    for(int i=1;i<=n;i++) Insert(i+n+n);

    int ans=0;
    while(q--){
        int op,x,y;
        cin>>op>>x>>y;
        x=(x+T*ans)%n+1,y=(y+T*ans)%n+op;
        if(op==0){
```

```
            Erase(x+n);
            a[x+n]=y;
            Insert(x+n);
        }else{
            if(x>y) swap(x,y);
            ans=Query(x+n,y+n+1);
            cout<<ans<<endl;
        }
    }

    return 0;
}
```