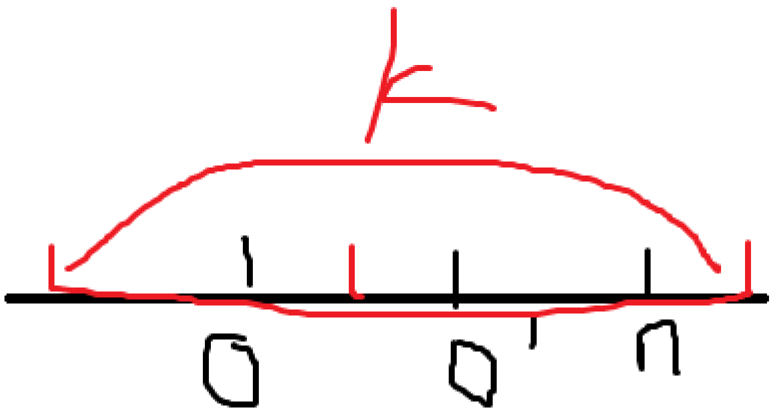
11.15 总结

	预估	实际
T1 **Reverse**	60	94
T2 **Silhouette**	0	8
T3 **Seat**	0	8
T4 **Ancient**	40	5

T1 reverse

想的是$O(nk)$ 的BFS，大体方向没问题，在纸上算了一下两个方向上对于 k 的范围进行区间翻转的要求

$eg.1$



此时对与O翻转到O`是不合法的，因为所需的k已经超过了限制，算是一个小剪枝，但在k很大的时候可以排除很多情况；

考虑如何对称，发现与k的奇偶性有关，所以可以预处理出，k对应的可翻转位置，在BFS时，加入新的节点，对于每个节点，先判断是否可以进行长度为k的翻转，在从合法的最近的翻转开始，依次加入预处理可以对称的点，并更新步数，最后T了两个点，卡常卡不过，只有写正解

正解　同样的BFS，但是题解很妙，省去了一系列复杂剪枝，可以$O(1)$ 找到最近的枚举点，然后在 $(k/2)$ 的拓展，加入下一个点，并且保证了每个点只会走到一次

Code

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 100010;
int n,m,k,s;
int dp[maxn],ban[maxn];
set<int>ji,ou;

template<typename T>void read(T &x)
{
    x=0; char c=getchar(); T neg=0;
    while(!isdigit(c)) neg|=!(c^'-'),c=getchar();
    while(isdigit(c)) x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(neg) x=(~x)+1;
}
template<typename T>void wr(T x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar((x-x/10*10)^48);
    return ;
}


int ckl(int st,int to,int k)
{
    if(st>=1) return st;
    return 1+(1-st);
}
int ckr(int st,int to,int k)
{
    if(st<=n) return st;
    return n-(st-n);
}
queue<int>q;
void BFS()
{
    q.push(s); dp[s]=0;
    while(q.size())
    {
        int u=q.front(); q.pop();
        int L=max(1,u-k+1), R=min(n,u+k-1);// 找到最远的拓展店
        L=L+(L+k-1)-u; R=R+(R-k+1)-u;// 找到合法的最近 / 远的拓展店
        if(L&1)
        {
            for(auto i=ji.lower_bound(L);i!=ji.end()&&*i<=R;ji.erase(i++))///注意
erase的顺序，在写的时候建议手写用 vis 进行标记
            {

                if(*i==s||ban[*i]) continue;
                dp[*i]=dp[u]+1; q.push(*i);
            }
        }
        else
```

```
        {
            for(auto i=ou.lower_bound(L);i!=ou.end()&&*i<=R;ou.erase(i++))/////
            {
                if(*i==s||ban[*i]) continue;
                dp[*i]=dp[u]+1; q.push(*i);
            }
        }
    }
}
signed main()
{
//  freopen("reverse.in","r",stdin);
//  freopen("reverse.out","w",stdout);
    read(n); read(k); read(m); read(s);

    for(int i=1;i<=n;i++)
    {
        if(i&1) ji.insert(i);
        else ou.insert(i);
    }
    int a;
    memset(dp,-1,sizeof dp);
    for(int i=1;i<=m;i++) read(a),ban[a]=1;
    BFS();
    for(int i=1;i<=n;i++) wr(dp[i]),putchar(' ');
    return 0;
}
```

T2 Silhouette

毒瘤，**毒瘤题**。 ~~虽然但是我看懂这篇题解花了5小时~~

首先，显然对于这道题，我们要求以下方程的解的个数： $\forall \in [1,n], \max\limits_{j=1}^{n} x_{i,j} = A_i, \max\limits_{j=1}^{n} x_{j,i} = B_i$

然后就是神奇操作，**将A,B进行排序**，因为这样对结果没有影响，简单证明一下： 先来考虑列，对于每一个A_i，无论哪一列在前，哪一列在后，最终所有列在第i行的最大值都需要是A_i,也就是说，X_{maxA} 在这一行里面可以随意移动，在列方向上也是如此，

如果最大的A_i不等于最大的B_i，那么无解，否则一定有解。 然后我们从大到小枚举A, B中每一个值S，对于排好序后每一个S，它们会是这样的：

接下来就是**分析图像**，类似分成了一层一层的。
然后我们枚举每一层。 先说S_1这一层，对于S_1有一个特殊性质，S_1是所有S中最大的，先设S_1所涉及到的这个矩形的长宽分别为a, b，

如下图：

我们在设$f[i]$表示a行中，**至少**有i行一定不合法的方案数，这里说的不合法是指高度没有到达S，也就是说不能对投影做贡献；之所以我们要设为至少，是因为这样每两行之间就可以不互相影响了。

先给逝子 $f[i] = C_a^i \times (S^i \times ((S+1)^{a-i} - S^{a-i}))^b$

解释

C_a^i　　在 a 中选 i 个位置不合法的方案数

S^i　　在该列中，这 i 个位置有 $[0, S-1]$ S 种取值 ，保证**至少**有 i 行不合法

$(S+1)^{a-i} - S^{a-i}$　　在该列中，剩余的 $a-i$ 个 位置可以有 $[0, S]$ $S+1$ 种取值，但是**至少要有 S 这个值存在**，因此减去 S^{a-i}

$()^b$　　有 b 列，自然是 b 次方

而我们要求的是 **恰好** 0 行不合法，

接下来就是**容斥**，$f[0]$ 是我们求出来的全集，观察 $f[1]$ 是如何得来的，在 C_a^1 时，我们枚举了所有 1 行不合法的情况

但是观察 $(S+1)^{a-i} - S^{a-i}$ ，会发现存在 1 行之外的**其他行**不合法，就会导致，在进行 C_a^1 的 “1 行不合法” 时

$(S+1)^{a-i} - S^{a-i}$的**其他行**重复了**之前** C_a^1 的 “1 行不合法”，就会导致有多余重复的情况，这部分就是我们多减的部分，因此需要加上 $f[2]$

以此类推，就是**容斥** ，即 $res = \sum\limits_{i=0}^{a} (-1)^i \times f[i]$

当然，这还**只是特殊情况**

考虑图一中的 S_2 一类的一般情况

因为每当我们处理完一个S之后，下一个区域的形状只有L行或矩形，如下图（红色为上一次处理的区域，紫色为这一次处理的区域）：

而无论对于哪种情况，我们都按如下划分方式将其划分为两部分：

对于矩形的两种情况，无非就是没有了那一部分，可以认为举行是一种特殊的L行。

为方便，我们不妨将L行区域按如下图方式标号a, b, c, d：

那么，现在我给出一般状态转移方程：
$f[i] = C_a^i \times (S^i \times ((S+1)^{a+c-i} - S^{a+c-i}))^b \times (S^i \times (S+1)^{a-i})^d$ 现在来解释这个更复杂的式子，先明确一下，因为上图中红色区域已经处理完毕，所以对于蓝色区域，其已经满足行，而没有满足列，对于绿色区域同理。

先来解释式子中的组合数C_a^i，你可能会存在疑问，为什么有$a+c$行，而不是C_{a+c}^i；那是因为上面我也明确过了，对于c行，行已经满足，我们不能让它不满足，而这i行的不满足我们只能从a行出；所以是C_a^i，而不是C_{a+c}^i。

再来解释$(S^i \times ((S+1)^{a+c-i} - S^{a+c-i}))^b$，对于$b$次方和$S^i$，上面对于$S$是最大的值的情况已经做了解释，在此就不做过多的赘述；直接解释$(S+1)^{a+c-i} - S^{a+c-i}$，这个也很好理解，与上面情况类似，有$i$行不合法，那么还有$a+c-i$个位置可以合法，而我们要保证这一列一定合法，所以还要减去不合法的情况。

最后来看后面新填进来的这一部分，应该能看的出来，这部分是为了处理矩形$a \times d$的，也就是区域2，式子与前面大体类似，不再赘述，只来思考这样一个问题：网上有另一篇题解（在我写之前可能也只有那一篇，而我就是按照那一篇的思路调出来的），他的后面这部分是$S^i \times (S+1)^{a-i} - S^{a-i})^d$，但是我的式子里却没有$-S^{a-i}$，他的式子的漏洞就在这里，而为什么不用这部分呢？

因为我们减去S^{a-i}是为了让那一列合法，还是开始我明确的那个问题，因为我们在处理红色区域的时候已经保证了其合法，而红色区域的S要比矩形$a \times d$（绿色区域）的大，也就是高，所以无论如何其正视图都已经不变化了，变化的只有左视图，而我们需要做的就是让左视图合法，不用考虑列合不合法的问题，所以不用减去S^{a-i}。

对于统计合法方案数的容斥，其式子亦是：$res = \sum\limits_{i=0}^{a}(-1)^i \times f[i]$ 对于$c = 0$和$d = 0$的情况，我们将其带入上式会发现对结果并无影响，于是我们可以将特殊情况的式子和普通情况的式子合并成一个。这样我们从达到小不断枚举每一个S即可求出所有合法方案数。

时间复杂度：$O(nlogn)$。

Code

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int maxn = 200010;
const int mod = 1e9+7;
int n,a[maxn],b[maxn],s[maxn*2+10],Ans=1;
int jc[maxn],invjc[maxn],inv[maxn];


template<typename T>void read(T &x)
{
    x=0; char c=getchar(); T neg=0;
    while(!isdigit(c)) neg|=!(c^'-'),c=getchar();
    while(isdigit(c)) x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(neg) x=(~x)+1;
}
template<typename T>void wr(T x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar((x-x/10*10)^48);
    return ;
}


void init()
{
    inv[1]=inv[0]=jc[1]=jc[0]=invjc[1]=invjc[0]=1;
    for(int i=2;i<=100010;i++) jc[i]=jc[i-1]*i%mod, inv[i]=(mod-
1ll*mod/i*inv[mod%i]%mod)%mod;
    for(int i=2;i<=100010;i++) invjc[i]=invjc[i-1]*inv[i]%mod;
```

```
}

int ksm(int a,int n)
{
    int ans=1;
    while(n)
    {
        if(n&1) ans=ans*a%mod;
        a=a*a%mod; n>>=1;
    }
    return ans;
}
int C(int x,int y){ return x<y?0:jc[x]*(invjc[x-y]*invjc[y]%mod)%mod; }
int lucas(int x,int y)
{
    if(x<y) return 0;
    if(!x) return 1;
    return lucas(x/mod,y/mod)*C(x%mod,y%mod);
}
int calc(int a,int b,int c,int d,int s)
{
    int ans=0,tmp;
    for(int i=0;i<=a;i++)
    {
        int tmp=( (  lucas(a,i) * (  ksm( ksm(s,i)*(ksm(s+1,a+c-i)-ksm(s,a+c-
i)+mod)%mod , b)  )  )%mod   )  * ksm(ksm(s,i)*ksm(s+1,a-i)%mod,d)
%mod; /////***********************
        if(!(i&1)) ans=(ans+tmp)%mod;
        else ans=(ans-tmp+mod)%mod;
    }
    return ans;
}
signed main()
{
//  freopen("Silhouette.in","r",stdin);
//  freopen("Silhouette.out","w",stdout);
    read(n);    init();

    for(int i=1;i<=n;i++) read(a[i]),s[i]=a[i];
    for(int i=1;i<=n;i++) read(b[i]),s[n+i]=b[i];


    sort(a+1,a+n+1);
    sort(b+1,b+n+1);
    if(a[n]!=b[n]){puts("0");return 0;}
    sort(s+1,s+2*n+1);

    s[0]=unique(s+1,s+2*n+1)-s-1;

//  cout<<s[0]<<endl;
    int nowa=n,nowb=n,prea=n+1,preb=n+1,va=a[n],vb=b[n];
    for(int i=s[0];i;i--)
    {
        while(nowa-1&&s[i]==a[nowa-1]) nowa--;
        while(nowb-1&&s[i]==b[nowb-1]) nowb--;
```

```
        Ans=Ans*calc(prea-nowa,preb-nowb,n-prea+1,n-preb+1,s[i])%mod;
        prea=nowa; preb=nowb;
    }
    wr(Ans);
    return 0;
}
/*
3
3 1 3
2 3 2



*/
```

T3 Seat

管他那么多干嘛，先鸽

T4 Ancient

怎么说呢，当然是**数论全家桶**

考场上以为可以 $40pts$ 的纯 $lucas$ 定理，但是挂了

正解

说人话

给定 $q,n(1\leq q,n\leq 10^9)$,计算

$$q^{\sum_{d|n}C_n^d} mod 999911659$$

若 $q=999911659$,则上式结果为 0 。否则，以为 $q=999911659$ 是质数，所以 q,n 互质。由欧拉定理的推论得

$q^{\sum_{d|n}C_n^d}=q^{\sum_{d|n}C_n^d mod 999911658}(mod\ 999911659)$

尝试分解质因数，发现 $999911658=2\times 3\times 4679\times 35617$

枚举 n 的正约数 d ,运用 $luacas$ 定理求组合数 C_n^d ，分别计算 $\sum_{d|n}C_n^d$ 对上述四个质数取模的结果，记为 a_1,a_2,a_3,a_4 。

使用 **中国剩余定理** 计算

$x\ mod\ 2=\ a_1$

$x\ mod\ 3=\ a_2$

$x\ mod\ 4679=\ a_3$

$x\ mod\ 35617=\ a_4$

即可得到 x 的最小整数解，使用快速幂求 q^x

证明：

令 $\sum_{d|n} C_n^d = a$，$999911658 = 2 \times 3 \times 4679 \times 35617 = P = p1 \times p2 \times p3 \times p4$

则有

$a = k \times (p1 \times p2 \times p3 \times p4) + x$

$a = k_1 \times p1 + a_1$

则有 $x \ mod \ p1 = \ a_1$

同理 $p2 \ p3 \ p4$

```
#include<bits/stdc++.h>
#define int long long
using namespace std;

const int mod=999911658,N=3e6+10;

template<typename T>void read(T &x)
{
    x=0; char c=getchar(); T neg=0;
    while(!isdigit(c)) neg|=!(c^'-'),c=getchar();
    while(isdigit(c)) x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(neg) x=(~x)+1;
}
template<typename T>void wr(T x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar((x-x/10*10)^48);
    return ;
}

int n,G,cnt,jc[N],inj[N];
int b[]={0,2,3,4679,35617 };


int ksm(int a,int n,int p)
{   int ans=1;
    while(n)
    {
        if(n&1) ans=(ans*a)%p;
        a=(a*a)%p; n>>=1;
    }
    return ans;
}
void init(int x)
{
    jc[0]=jc[1]=1;
    for(int i=2;i<=x;i++) jc[i]=jc[i-1]*i%x;
```

```
}

int C(int x,int y,int p) {return (x<y)?0:1ll*jc[x]*ksm(jc[x-y]*jc[y],p-2,p)%p;}

int lucas(int x,int y,int p)
{
    if(x<y) return 0;
    if(!x) return 1;
    return lucas(x/p,y/p,p)*C(x%p,y%p,p)%p;
}
int m[N],a[N];
int crt()
{
    int ans=0;
    for(int i=1;i<=4;i++)
    {
        m[i]=mod/b[i];
        ans=(ans+(a[i]*m[i]%mod)*ksm(m[i],b[i]-2,b[i])%mod)%mod;
    }
    return ans%mod;
}
signed main()
{
    freopen("Ancient.in","r",stdin);
    freopen("Ancient.out","w",stdout);
    read(n); read(G);
    for(int i=1;i<=4;i++)
    {
        init(b[i]);
        for(int j=1;j*j<=n;j++)
        {
            if(n%j) continue;
            a[i]=(a[i]+lucas(n,j,b[i]))%b[i];
            if(j*j!=n) a[i]=(a[i]+lucas(n,n/j,b[i]))%b[i];
        }
    }
    cnt=crt();
//  cout<<cnt<<endl;
    wr(ksm(G,cnt,mod+1));
    return 0;
}
```