11月15号的题解：

期望得分：

题目：	期望	实际
Reverse	100 =v=	100 =v=
Silhouette	16	16
Seat	8	8
Ancient	nan	nan
总分	124	124

T1:

普通的bfs是平凡的（然而可以直接卡过去），但可能会被卡满Nk，原因是每个位置必须扩展满n次。

考虑减少枚举次数，如果能在一个点被访问之后直接弹出它就好了，这就是set做的事情。

进一步考虑题目的性质，发现：

1. 一个点所扩展的点的奇偶性是确定的，
2. 它能够扩展的点也是在一段区间里的数。
3. 一个点只会被扩展一次。所以进一步启发我们使用set作为动态数组。

那么做法已经很明显了。分别用两个set记录未被访问的奇数和偶数，每次取出可被修改的数，更新并弹出set即可。

```
#include<bits/stdc++.h>
using namespace std;
#define R(i) for(int i=1; i<=n; ++i)
#define Rep(i,x,y) for(int i=x;i<=y;++i)
#define pii pair<int, int>
#define mp make_pair
#define Rg(i,x,y,G) for(int
i=G.head[x],y=G.e[i].to;i;i=G.nex[i],y=G.e[i].to)
const int N=1e5+10;
int n,k,m,s;
int a[N], c[N], f[N];
void prt(int x) {
    if(x<0) {putchar('-'); x=-x;}
    if(x<10) {putchar(x^'0'); return;}
    prt(x/10), putchar((x%10)^'0');
}
bool jud(int z, int k) {
    if(k&1) return (z-k/2>=1)&&(z+k/2<=n);
    else return (z-k/2+1>=1)&&(z+k/2<=n);
}
set<int> S[2];
typedef set<int>::iterator sit;
```

```cpp
int main() { int x,y,z,tot=1,l,r,h,L,R;
freopen("reverse.in", "r", stdin);
freopen("reverse.out", "w", stdout);
    memset(f, -1, sizeof f);
    scanf("%d%d%d%d",&n,&k,&m,&s); h=!(k&1);
    Rep(i,1,m) scanf("%d",a+i), c[a[i]]=1,++tot;
    queue<int> q;
    q.push(s); f[s]=0;
    R(i) if(!c[i]&&s!=i) S[i&1].insert(i);
    while(q.size()) {
        x=q.front(); q.pop();
        L=max(1,x-k+1); R=min(n,x+k-1);
        if(S[(x&1)^h].size()==0) continue;
        l=max(1, L+(L+k-1-x)), r=min(n, R-(x-(R-k+1)));
        sit i=S[(x&1)^h].lower_bound(l);
        for(sit j=i;l<=*i&&*i<=r&&i!=S[(x&1)^h].end();i=j) {
            j=i; ++j;
            q.push(*i); f[*i]=f[x]+1;
            S[(x&1)^h].erase(i);
        }
    }
    R(i) prt(f[i]), putchar(' ');
    return 0;
}

```

T2

先转化题意为：a，b 数组是每行每列的最大值。

由于限制值不同，非常难以处理，考虑先归化成相同限制值：

把 a , b 以降序排序，并把每一个值丢进值域数组D中

枚举`D[i]`每次取出等于`D[i]`的行列：

第一次取出时，他们组成一个矩形，现在考虑这些点的方案数：即a行b列限制都为s的方案

考虑容斥，用 f(i) 表示：

至少 i 行不存在等于 w 的数，且每一列都有等于 w 的数的方案数。（取值范围为`0 ~ w`）

那么

$$f(i) = C_a^i \times ((w+1)^{a-i} - w^{a-i}) \times (w^i))^b$$

- 解释一下： C 是指 a 行中选择 i 行不满足条件。
  - 而对于每一列来说：
    - $((w+1)^{a-i} - w^{a-i})$ 表示所有可能情况减去没有出现w的情况，即这一列至少出现一个 w。
    - (w^i) 表示有 i 行只出现 `0 ~ w-1` ，没有w的情况。
  - 最后 b 次方表示一共 b 列。

```cpp
#include<bits/stdc++.h>
using namespace std;
#define R(i) for(int i=1; i<=n; ++i)
#define Rep(i,x,y) for(int i=x;i<=y;++i)
#define pii pair<int, int>
#define mp make_pair
#define Rg(i,x,y,G) for(int i=G.head[x],y=G.e[i].to;i;i=G.nex[i],y=G.e[i].to)
#define int long long
const int N=4e5+10,p=1e9+7;
int n, t;
int a[N],b[N],d[N];
int fc[N], rf[N];
int fx(int x) { return (x%p+p)%p; }
int qpow(int a, int n=p-2) {
    int c=1; n=(n%(p-1)+(p-1))%(p-1);
    while(n) {
        if(n&1) c=(1ll*c*a%p);
        a=1ll*a*a%p;
        n>>=1;
    } return c;
}
bool cmp(int a, int b) { return a>b; }
int cal(int la,int lb,int a,int b,int w) {
    int res=0;
    for(int i=0; i<=a; ++i) {
        int ret=1;
        ret=1ll*ret*qpow(fx(qpow(w+1, la+a-i)-qpow(w,la+a-i)), b)%p;
        ret=1ll*ret*qpow(w+1, 1ll*(a-i)*lb)%p*qpow(w, 1ll*i*(lb+b))%p;
        ret=1ll*ret*rf[a-i]%p*rf[i]%p;
        if(i&1) res=fx(res-ret); else res=(res+ret)%p;
    } res=1ll*res*fc[a]%p;
    return res;
}

signed main() {
    freopen("silhouette.in","r",stdin);
    freopen("silhouette.out","w",stdout);
    fc[0]=1; Rep(i,1,N-1) fc[i]=1ll*fc[i-1]*i%p;
    rf[N-1]=qpow(fc[N-1]); for(int i=N-2;i>=0;--i) rf[i]=1ll*rf[i+1]*(i+1)%p;
    scanf("%lld", &n);
    R(i) scanf("%lld",a+i),d[++t]=a[i];
    R(i) scanf("%lld",b+i),d[++t]=b[i];
    sort(a+1,a+1+n,cmp);
    sort(b+1,b+1+n,cmp);
    sort(d+1,d+1+t,cmp);
    int la=0, lb=0, ans=1;
    for(int i=1;i<=t;++i) { int ra=la, rb=lb;
        while(d[i]==a[ra+1]&&ra<n) ++ra;
        while(d[i]==b[rb+1]&&rb<n) ++rb;
        ans=(1ll*ans*cal(la,lb,ra-la,rb-lb,d[i]))%p;
        if(ans==0) break;
        la=ra, lb=rb;
    }
    printf("%lld", ans);
    return 0;
}
```

```
57
```

T4

引理：

1.

$$a^x \equiv a^{x \bmod p-1} (\bmod p)$$

2.

$$Ans = G^{\sum_{d|n} C_n^{n/d}} \bmod p$$

3.

若

$x \equiv a_1 (\bmod b_1)$

$x \equiv a_2 (\bmod b_2)$

$x \equiv a_3 (\bmod b_3)$

$x \equiv a_4 (\bmod b_4)$

且$b_1 \times b_2 \times b_3 \times b_4 = p$

那么根据 `excrt`，可以求出 a 在 mod p 意义下的值。

观察题目，发现 p-1 不是质数，那么我们不能直接统计幂次，

考虑分解质因数：`p-1 = 2*3*4679*35617`

根据上述引理，可以发现可以分别统计在模每一个质因数意义下的幂次和，最后普通`crt`合并即可。

```cpp
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define R(i) for(int i=1; i<=n; ++i)
#define Rep(i,x,y) for(int i=x;i<=y;++i)
#define pii pair<int, int>
#define mp make_pair
#define Rg(i,x,y,G) for(int
i=G.head[x],y=G.e[i].to;i;i=G.nex[i],y=G.e[i].to)
const int p=999911658;
int n,G,fc[50010],a[5],b[5]={0,2,3,4679,35617},val;
int qpow(int a, int n, int p) { int ret=1;
    for(;n;n>>=1,a=1ll*a*a%p)
        ret=ret*(n&1 ? a:1)%p;
    return ret;
}
int rev(int a, int p) { return qpow(a,p-2,p); }
void init(int p) {
    fc[0]=1; Rep(i,1,p) fc[i]=fc[i-1]*i%p;
}
int C(int n,int m,int p) {
    if(n<m) return 0;
```

```
    return fc[n] * rev(fc[m],p)%p * rev(fc[n-m],p)%p;
}
int luc(int n,int m,int p) {
    if(n<m) return 0;
    if(!n) return 1;
    return C(n%p, m%p, p) * luc(n/p, m/p, p)%p;
}
int crt() { int c=0;
    Rep(i,1,4) c=(c+ a[i]*(p/b[i])%p*rev(p/b[i],b[i])%p)%p;
    return c;
}
signed main() {
    freopen("ancient.in","r",stdin);
    freopen("ancient.out","w",stdout);
    scanf("%lld%lld",&n,&G);
    if(G%(p+1)==0) return 0&printf("0\n");

    Rep(k,1,4) { init(b[k]);
        for(int i=1;i*i<=n;++i) {
            if(n%i==0) {
                a[k]=(a[k]+luc(n,i,b[k]))%b[k];
                if(i*i!=n) {
                    a[k]=(a[k]+luc(n,n/i,b[k]))%b[k];
                }
            }
        }
    }
    printf("%lld", qpow(G,crt(),p+1));
    return 0;
}


```

总结：

1. 记住以往的技巧。
2. 交代清楚自己的作息。
3. 全力以赴。

以前最喜欢的壁纸：