20221115测试总结

题解

reverse

题意

给出一个 01串，初始时只有给定的位置 S 为 1，其余位置均为 0。同时有 m 个禁区，禁区始终不可以为 1。现在你可以选定一段连续的 k 段子串，并将其镜像反转。试问每一个点，其为 1 的所需操作数。

解答

我们对于每个点，每次都找到可以让其通过上述操作变成 1 的其他点。这样一来，我们将他们连上一条边，进而变成从 S 开始的单源最短路径问题。直接建边时间复杂度达到 $O(n^2)$，不可承受。接下来考虑优化。

对于每个点，每次都定好其选择的区间 $[L,R]$，这个区间内如果 L 为偶数，那么区间内都会选择偶数，否则奇数。故用两个 `set` 分别记录偶数和奇数。之后跑一个 bfs 即可。

```
#include <set>
#include <queue>
#include <bitset>
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <iostream>
#include <algorithm>
#define ll long long
using namespace std;
int n,k,m,s;
bitset<100005> vis;
set<int> odd,even;

inline ll re(){
    ll k=0,f=1;
    char cre=getchar();
    while(!('0'<=cre&&cre<='9')){
        if(cre=='-') f=-1ll;
        cre=getchar();
    }
    while('0'<=cre&&cre<='9'){
        k=(k<<1ll)+(k<<3ll)+(cre^48ll);
        cre=getchar();
    }
    return 1ll*k*f;
}

void wr(ll x){
    if(x<0){
        putchar('-');
        x=~x+1;
    }
    if(x>9) wr(x/10ll);
```

```
    putchar(x%10ll^48ll);
}

int dis[100005];
inline void bfs(){
    queue<int> q;
    memset(dis,-1,sizeof(dis));
    dis[s]=0;
    q.push(s);
    while(q.size()){
        int x=q.front();
        q.pop();
        int l=max(1,x-k+1);
        int r=min(n,x+k-1);
        l=l+(l+k-1)-x;
        r=r+(r-k+1)-x;
        set<int> &sat=l&1?odd:even;
        for(auto i=sat.lower_bound(l);i!=sat.end()&&*i<=r;sat.erase(i++)){
            dis[*i]=dis[x]+1;
            q.push(*i);
        }
    }
}

signed main(){
    n=re(),k=re(),m=re(),s=re();
    for(int i=1;i<=m;++i) vis[re()]=1;
    for(int i=1;i<=n;++i){
        if(i!=s&&!vis[i])
            i&1?odd.insert(i):even.insert(i);
    }
    bfs();
    for(int i=1;i<=n;++i){
        wr(dis[i]);
        putchar(i!=n?' ':'\n');
    }
    return 0;
}
```

silhouette

题意

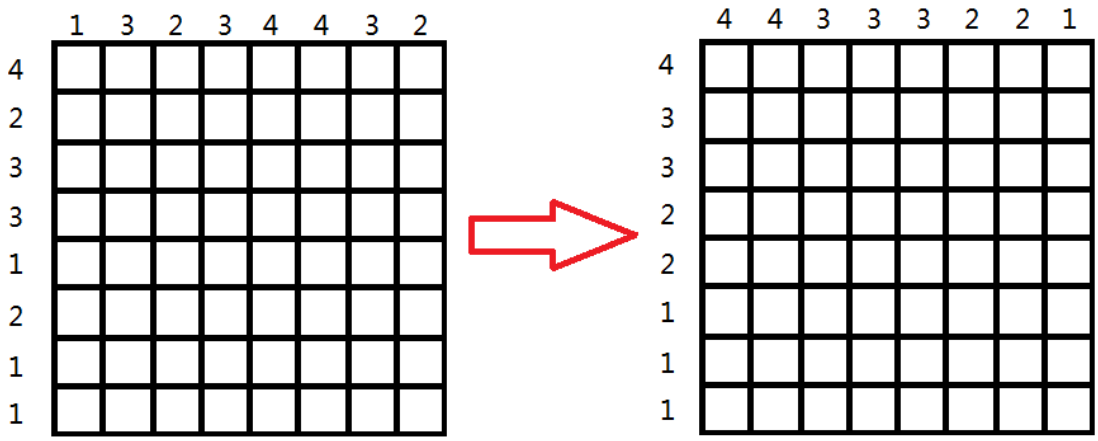
有一个 $n \times n$ 的网格，在每个格子上堆叠了一些边长为 1 的立方体。

现在给出这个三维几何体的正视图和左视图，求有多少种与之符合的堆叠立方体的方案。两种方案被认为是不同的，当且仅当某个格子上立方体的数量不同。

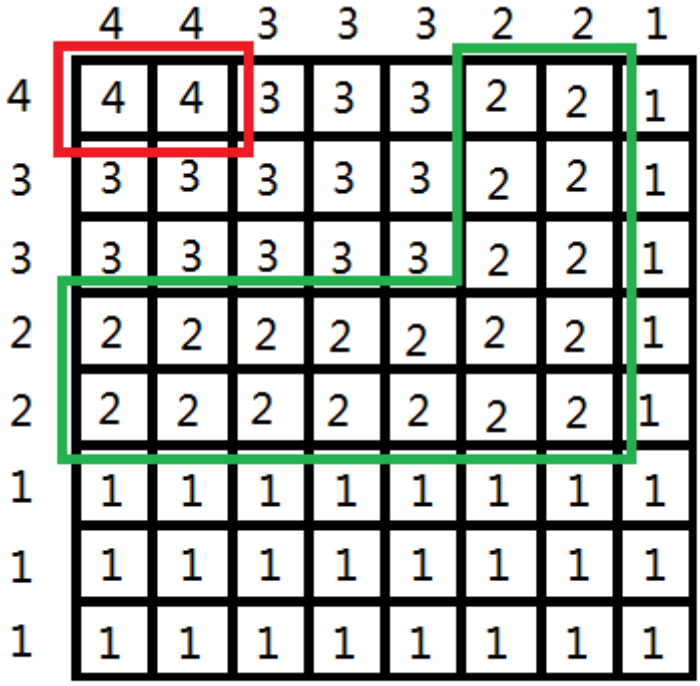
输入中将会给出 n 的大小以及正视图从左到右所看见的高度序列 A_i 和左视图从左到右所看见的高度序列 B_i。你需要输出方案数对 $10^9 + 7$ 取模的结果。

解答

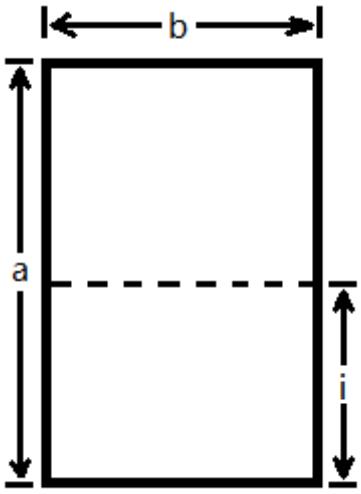
事实上如果将 A_i 和 B_i 都按同样的方式进行升序或者降序排序，那么并不会影响答案，如下图：



所以，我们将 A_i ，B_i 进行降序排序，每个方格 $x_{i,j}$ 所填的数就等于 $x_{i,j}=\min\{A_i,B_j\}$。如下图：



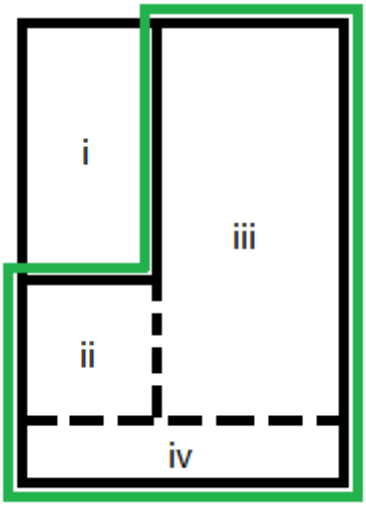
于是就会形成几个值全是 s （$s\le x_{i,j}$）的区域，矩形或者 L 形。接下来考虑计算这些区域对答案的贡献。首先考虑矩形。



如上图，我们对这样一个 $a \times b$ 的矩阵进行填数，显然每个数必须都是 s 才是合法方案。所以我们钦定 i 行不合法，这部分答案就为 $C_a^i \times s^{ib}$，在这里 s 表示 $0 \sim s-1$ 这 s 个数；而剩下的 $a-i$ 行都要合法，这部分答案就是 $[(s+1)^{(a-i)b} - s^{(a-i)b}]$，即选择 $0 \sim s$ 的方案数减去 $0 \sim s-1$ 的方案数就是选 s 这一个数的方案数。

所以令 $f(i) = C_a^i \times s^{ib} \times [(s+1)^{(a-i)b} - s^{(a-i)b}]$，这部分答案根据容斥原理可得：
$ans = \sum_{i=0}^{a} (-1)^i \times f(i)$。

L 形类似，只不过我们需要把它分成四个矩形，如下图：



其中 i 是我们已经计算过的矩形，iv 是钦定的不合法矩形，ii, iii 都是分割出的矩形。

剩下的计算类似于矩形。只是要注意 ii 矩形的计算，由于 i 一定合法，所以在对 ii 计算时只需要 $(s+1)$ 即可，式子推导留作习题。

seat

题意

有 $n+2$ 个座位等距地排成一排，从左到右编号为 0 至 $n+1$。

最开始时 0 号以及 $n+1$ 号座位上已经坐了一个小 G，接下来会有 n 个小 G 依次找一个空座位坐下。由于小 G 们坐得太近就容易互相搏弈，每个小 G 会找一个当前离最近的小 G 距离最远的座位坐下。如果有多个备选的座位，这个小 G 会等概率选择其中一个。

给出 n，求第 i 个坐下的小 G 坐在 j 号座位的概率, 对给出的 P 取模。

解答

本人太菜，只解决了 $n = 2^x - 1$。

古代猪文

~~古代猪瘟~~

题意

给定整数 n, g $(1 \le n, g \le 10^9)$，试计算 $g^{\sum_{d|n} C_n^d} \bmod 999911659$。

解答

首先如果 $g \equiv 0 \pmod{999911659}$，则答案为 0。否则，由于 999911659 为质数，那么据欧拉定理有：

$$g^{\sum_{d|n} C_n^d} \equiv g^{\sum_{d|n} C_n^d \bmod 999911658} \pmod{999911659}$$

问题转为：求解 $\sum_{d|n} C_n^d \bmod 999911658$。

这里，我们分解质因数，可以得到 $999911658 = 2 \times 3 \times 4679 \times 35617$。之后枚举 n 的每个因数 d，利用 Lucas 定理来计算 C_n^d，分别计算 $\sum_{d|n} C_n^d$ 对上面四个质数取模后的结果 a_1, a_2, a_3, a_4。最终利用 CRT 计算线性同余方程组：

$$\begin{cases} x \bmod 2 = a_1 \\ x \bmod 3 = a_2 \\ x \bmod 4679 = a_3 \\ x \bmod 35617 = a_4 \end{cases}$$

最后答案就是 g^x。

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <iostream>
#include <algorithm>
#define ll long long
using namespace std;
ll val;
int n,g;
ll a[5];
ll fac[50005];
ll const p=999911658;
ll b[5]={0,2,3,4679,35617};

inline ll re(){
    ll k=0,f=1;
    char cre=getchar();
    while(!('0'<=cre&&cre<='9')){
        if(cre=='-') f=-1ll;
        cre=getchar();
    }
    while('0'<=cre&&cre<='9'){
        k=(k<<1ll)+(k<<3ll)+(cre^48ll);
        cre=getchar();
    }
    return 1ll*k*f;
}

void wr(ll x){
    if(x<0){
        putchar('-');
        x=~x+1;
    }
    if(x>9) wr(x/10ll);
    putchar(x%10ll^48ll);
}
```

```cpp
inline void init_(ll mod){
    fac[0]=1;
    for(int i=1;i<=mod;++i)
        fac[i]=1ll*i*fac[i-1]%mod;
}

inline ll qpow(ll a,ll b,ll mod){
    ll res=1;
    for(;b;b>>=1){
        if(b&1) res=1ll*res*a%mod;
        a=1ll*a*a%mod;
    }
    return res;
}

inline ll C(ll n,ll m,ll mod){
    return n<m?0:1ll*fac[n]*qpow(fac[n-m]*fac[m],mod-2,mod)%mod;
}

ll Lucas(ll n,ll m,ll mod){
    if(n<m) return 0;
    if(!n) return 1;
    return 1ll*Lucas(n/mod,m/mod,mod)*C(n%mod,m%mod,mod)%mod;
}

inline void CRT(){
    for(int i=1;i<=4;++i)
        val=(val+a[i]*(p/b[i])%p*qpow((p/b[i]),b[i]-2,b[i])%p)%p;
}

signed main(){
    n=re(),g=re();
    if(g==p+1||!g){
        putchar(48);
        return 0;
    }
    for(int i=1;i<=4;++i){
        init_(b[i]);
        for(int j=1;j*j<=n;++j){
            if(n%j==0){
                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];
            }
        }
    }
    CRT();
    wr(qpow(g,val,p+1));
    return 0;
}
```