T435167 01 Sort

无解当且仅当无序且同色，不能进行任何操作。

否则，考虑3个位置 $(i,0),(j,0),(k,1)\to(k,1),(j,0),(i,0)\to(j,0),(k,1),(i,0)\to(j,0),(i,0),(k,1)$，这样就完成了交换 i,j 操作，那么依次让每个数归位即可。

拓展：

1. 最小化操作次数
2. m 种颜色，最小化操作次数

U76034 拆分数列计数

将 x 进行质因子分解 $x=\prod p_i^{e_i}$，把 e_i 个 p_i 分配到 n 份中、可以为空，应用插板法方案数为 $\binom{e_i+n-1}{e_i-1}$，这里 $n=10^{16},e_i\le 20$，使用按列递推的方式求组合数。

对于正负号，贡献是 2^{n-1}（前 $n-1$ 个正负任选，最后调回来）。

```
#include<bits/stdc++.h>
#define MAXN 100005
#define P int(1e9+7)
using namespace std;

typedef long long ll;

int T,x;
ll n, ans = 1;
ll qpow(ll x, ll n){
    ll res = 1;
    while(n){
        if(n&1) res = res * x % P;
        x = x * x % P;
        n /= 2;
    }
    return res;
}

ll comb(ll n, int k){
    n %= P;//注意这里要预先%P
    k %= P;
    ll res = 1;
    for(int j=1;j<=k;j++){
        res = res * (n-j+1+P) % P * qpow(j, P-2) % P;
    }
    return res;
```

```cpp
}

void div(int x){
    int cnt = 0;
    for(ll i=2;i*i<=x;i++){
        cnt = 0;
        while(x%i==0){
            x /= i;
            ++cnt;
        }
        if(cnt > 0) ans = ans * comb(cnt+n-1, cnt) % P;
    }
    if(x > 1) ans = ans * comb(n, 1) % P;
}


int main(){
    cin>>T;
    while(T--){
        cin>>x>>n;
        ans = 1;
        div(x);
        cout<<ans * qpow(2,n-1) % P<<'\n';
    }
    return 0;
}
```

U287193 冒泡

对于排列 p:

- $pos[x]$ 代表 x 的位置，即 $p[pos[x]] = x$
- $inv[x] = x$ 之前 $> x$ 的数字数量，即 $inv[x] = \sum_{i=1}^{pos[x]} [p[i] > x]$，不同的 $inv[x]$ 和 p 一一对应

求 $f(a, m)$ 的方法：设初始排列为 a'，对应 inv'，经过 m 轮之后变为 a，对应 inv。由于一一对应，可以对 inv' 计数求 $f(a, m)$:

- 首先必须满足 $a[i] = i, i > n - m$（也就是 $inv[x] = 0, x > n - m$），否则无解
- 若 $inv[x] > 0$，那么 $inv'[x] = inv[x] + m$
- 若 $inv[x] = 0$，那么 $inv'[x] \in [0, m]$
- 同时 $inv'[x] \le n - x$

先判无解，若有解 $f(a, m) = (m + 1)^t m!$，这里 $t = \sum_{i \le n-m} [inv[a[i]] = 0]$，也就是 $a[1 \sim n - m]$ 中前缀最大值的个数。到这里可以通过子任务2，获得50分。

接下来对 $a[1 \sim n]$ 进行带权计数，由于 $f(a, m)$ 仅和其前缀最大值的个数有关，设 $dp[i][j]$ 表示前 i 个数、当前前缀最大值为 j 的答案

如果 $lim[i] = 0$:

- 第 i 位填最大值 j，此时增加1个前缀最大值：$(m+1)\times\sum_{k<j} dp[i-1][k] \to dp[i][j]$
- 第 i 位填 $v<j$：$dp[i-1][j]\times(j-i+1-cnt[i+1][j-1]) \to dp[i][j]$，这里 $cnt[i][j]$ 代表 $lim[i\sim n]$ 中已经确定 $\le j$ 的数量，即 $\sum_{k\ge i}[lim[k]\le j]$

如果 $lim[i]\neq 0$：

- 第 i 位填最大值 j，则要求 $lim[i]=j$：$(m+1)\times\sum_{k<j} dp[i-1][k] \to dp[i][j]$
- 第 i 位填 $lim[i]<j$：$dp[i-1][j] \to dp[i][j]$

使用前缀和优化转移，复杂度 $O(n^2)$。

```cpp
#include<bits/stdc++.h>
#define MAXN 5005
#define mod 998244353
using namespace std;
typedef long long ll;

int n,m;
int lim[MAXN];
int vis[MAXN], cnt[MAXN][MAXN];
ll fac[MAXN], dp[MAXN][MAXN], sdp[MAXN];

void add(ll a, ll& b) { b = (a + b) % mod;}//a->b
int main(){
    cin>>n>>m;
    fac[0] = 1;
    for(int i=1;i<=n;i++) fac[i] = fac[i-1] * i % mod;

    int ok = 1;
    for(int i=1;i<=n;i++){
        cin>>lim[i];
        int x = lim[i];
        if(x && i>n-m) ok &= (lim[i]==i);
        if(x) vis[x] = 1, cnt[i][x] = 1;
    }
    if(ok == 0){//无解
        cout<<"0";
        return 0;
    }

    for(int i=n;i>=1;i--){//二维前缀和
        for(int j=1;j<=n;j++){
            cnt[i][j] += cnt[i][j-1] + cnt[i+1][j] - cnt[i+1][j-1];
        }
    }

    n -= m;//最后m个数inv=0
    dp[0][0] = 1;
    for(int i=1;i<=n;i++){
        sdp[0] = dp[i-1][0];
        for(int j=1;j<=n;j++) sdp[j] = (sdp[j-1] + dp[i-1][j]) % mod;
        for(int j=1;j<=n;++j){
```

```
            if(lim[i] == 0){
                if(vis[j] == 0) add(sdp[j-1] * (m+1) % mod, dp[i][j]);//1
                add(dp[i-1][j] * (j-i+1-cnt[i+1][j-1]) % mod, dp[i][j]);//2
            }
            else{
                if(lim[i] == j) add(sdp[j-1] * (m+1) % mod, dp[i][j]);//3
                if(lim[i] < j) add(dp[i-1][j], dp[i][j]);//4
            }
        }
    }
    cout<<dp[n][n]*fac[m]%mod;
    return 0;
}
```

P9371 [APIO2023] 序列 / sequence

子任务2 $n \le 2000$

枚举左端点 l 然后正推右端点 r，使用对顶堆的方式维护中位数+插入，用桶维护中位数出现次数。复杂度 $O(n^2 logn)$。

枚举左端点 l 然后反推右端点 r，使用链表+指针的方式维护中位数+删除，用桶维护中位数出现次数。复杂度 $O(n^2)$。

子任务3 序列先上升再下降

对于每个值 x，出现下标构成2个连续段，只用检查是否可以同时取到2段即可。

子任务4 $a[i] = 1, 2, 3$

首先检查 $a[1 \sim n]$ 的全局中位数 med：

1. 如果 $med \neq 2$，则其出现次数必然超过 $n/2$，肯定就是答案；
2. 否则 $x = 2$ 可以贡献其全部出现次数；

接下来考虑 $x = 1$（$x = 3$ 同理），构造另一个数组 b，$a[i] = 1$ 的位置标 1、$a[i] \neq 1$ 的位置标 -1。则子段中位数能取到 1 当且仅当区间 b 的和 ≥ 0。不妨枚举 r，找到最靠前的 $s_b[l-1] \le s_b[r]$，这是二维偏序。

子任务5 每个数最多出现 2 次

检查答案是否可以是 2。

对于一个值 x，找到其出现位置 p_1, p_2，检查是否存在合法区间 $[l, r]$ 满足：

1. $l \le p_1 \le p_2 \le r$
2. x 是中位数

构造数组 b（$<x,=x,>x$ 的数分别标记为 $-1,0,+1$），那么区间 $[l,r]$ 合法当且仅当其区间和 $s\in[-2,+2]$，检查方法如下：

- 不妨假设 $s[p_1,p_2]<-2$，查找 $[1,p_1-1]$ 的最大后缀（sum_1）和 $[p_2+1,n]$ 的最大前缀（sum_2），如果 $sum_1+s[p_1,p_2]+sum_2\ge 2$ 则存在合法区间，这是因为区间边界在移动时区间和是连续变化的，也就是区间和值域范围里的所有值都能取到；如需构造的话可以使用双指针或者线段树二分可以找到合法区间

从小->大枚举 x，用类似区间最大子段和的方法维护数组 b（变化量 $O(n)$ 量级）。总复杂度 $O(nlogn)$。

子任务6&7

构造数组 H（$<x,\ge x$ 的数分别标记为 $-1,+1$），L（$\le x,>x$ 的数分别标记为 $-1,+1$）

- 区间合法的充要条件是 $H[l,r]\ge 0$ 且 $L[l,r]\le 0$
- 任意区间满足 $H[l,r]\ge L[l,r]$

使用类似子任务5的方式，对于一个值 x 的出现位置 p_1,p_2，检查是否存在合法区间（所不同的是其他位置也会出现 x）：

- 查询最大的 $v_1=H[l_1,r_1]$ 满足 $l_1\le p_1\le p_2\le r_1$
- 查询最小的 $v_2=L[l_2,r_2]$ 满足 $l_2\le p_1\le p_2\le r_2$
- 存在合法区间当且仅当 $v_1\ge 0$ 且 $v_2\le 0$。

> 考虑区间 $[l_1,r_1]$，假设 $L[l_1,r_1]>0$，那么在 $l_1\to l_2,r_1\to r_2$ 过程中必然存在 $[l,r]$ 使得 $L[l,r]=0$ 同时 $H[l,r]\ge 0$。

对于一个值的所有出现位置 $p_1,p_2,...,p_k$，如果对于 p_1,p_k 存在合法区间，那么对于 $p_1,p_{i<k}$ 都存在合法区间；利用单调性可以做双指针，总复杂度 $O(nlogn)$。

```
//检查是否存在合法区间[x,y]满足x<=l<=r<=y
bool chk(int l, int r){
    data x, y, z;
    x = query(rt1, 1, l-1), y = query(rt1, l, r), z = query(rt1, r+1, n);
    int v1 = x.df1 + y.s + z.dp1;//最大H
    x = query(rt2, 1, l-1), y = query(rt2, l, r), z = query(rt2, r+1, n);
    int v2 = x.df2 + y.s + z.dp2;//最小L
    return v1>=0 && v2<=0;
}

int solve(int x){//中位数=x
    for(int i:pos[x-1]) change(rt1, i);
    for(int i:pos[x]) change(rt2, i);

    int len = pos[x].size();
    int ans = 1;
    for(int k1=0,k2=0;k1<len;++k1){//双指针
        if(k2<k1) k2 = k1;
        while(k2+1<len && chk(pos[x][k1],pos[x][k2+1])) ++k2;
        ans = max(ans, k2-k1+1);
```

```
    }
    return ans;
}
```