Atcoder Rugular Contest 124

比赛链接

D - Yet Another Sorting Problem

题意

给定一个 $n+m$ 的排列，每次可以选取一个位置位于 $1\sim n$ 的元素和一个位置位于 $n+1\sim n+m$ 的元素交换位置。

问使得排列有序的最小操作次数。

题解

将排列分解成若干置换环，则对于每个交换操作，等价于选取 $1\le i\le n,n+1\le j\le n+m$，将 $i\to p_i,j\to p_j$ 变为 $i\to p_j,j\to p_i$。

不难发现如果 i,j 属于同一个置换环则环分裂，否则两个环合并。

同时将前 n 个位置染黑，后 m 个位置染白，于是每次操作需要选中一个黑点和一个白点进行操作。

设 A_i 表示第 i 次操作后大小至少为 2 的纯黑色环个数，B_i 表示第 i 次操作后大小至少为 2 的纯白色环个数，C_i 表示第 i 次操作后置换环个数。

定义势能函数 $f(i)=n+m-C_i+2\max(A_i,B_i)$，不难发现 $|C_{i+1}-C_i|=1,|A_{i+1}-A_i|\le 1,|B_{i+1}-B_i|\le 1$。

同时有 $(C_{i+1}-C_i)(A_{i+1}-A_i)\ge 0,(C_{i+1}-C_i)(B_{i+1}-B_i)\ge 0$，于是有 $f(i+1)\ge f(i)+1$。

终态 $A_z=B_z=0,C_z=n+m$，于是有 $f(Z)=0$，最小操作次数不小于 $f(0)-f(Z)=n+m-C_0+2\max(A_0,B_0)$。

下面证明下界可以取到：

首先如果 $A_i\gt 0,B_i\gt 0$，可以选取一个纯黑环和一个纯白环合并，这样 $A_{i+1}-A_i=B_{i+1}-B_i=C_{i+1}-C_i=-1, f(i+1)=f(i)-1$。

若 $A_i\gt 0,B_i=0$，可以选取一个纯黑环和一个含白点的环合并，这样 $\max(A_{i+1},B_{i+1})-\max(A_i,B_i)=-1,f(i+1)=f(i)-1$。

$A_i=0,B_i\gt 0$ 类似处理。最后考虑 $A_i=0,B_i=0$ 的情况，不妨任取一个大小不为 1 的置换环，假设环上黑点不少于白点。

可以找到 i,p_i 满足 i 是白点且 p_i 是黑点，交换位置 i,p_i 的元素等价于从环上单独拆出 p_i。

这样环上黑点数减一，但显然不会形成大小超过 1 的纯白环。于是 A,B 不变，$C_{i+1}=C_i-1,f(i+1)=f(i)-1$。

```
const int MAXN=1e5+5;
int n,m,a[MAXN<<1],vis[MAXN<<1];
int cnt1,cnt2;
void dfs(int pos){
    if(vis[pos])return;
    vis[pos]=true;
    if(pos<=n)cnt1++;
    else
    cnt2++;
    dfs(a[pos]);
}
int main(){
    n=read_int(),m=read_int();
    _rep(i,1,n+m)
    a[i]=read_int();
    int ans=n+m,s1=0,s2=0;
    _rep(i,1,n+m){
        if(!vis[i]){
            ans--;
            cnt1=cnt2=0;
            dfs(i);
            if(cnt1>=2&&cnt2==0)
            s1++;
            else if(cnt2>=2&&cnt1==0)
            s2++;
        }
    }
    enter(ans+2*max(s1,s2));
    return 0;
}
```

E - Pass to Next

题意

给定 n 个人围成一个环，第 i 个人初始时有 a_i 个球，接下来第 i 个人选中 $0\le k\le a_i$ 个球传给第 $i+1$ 个人。

注意第 n 个人把球给第一个人，且所有人同时传球。设一次操作后第 i 个人手上有 b_i 个球，则得到序列 B

对所有可能序列 B先进行一次去重，然后定义每个序列权值为 $\prod_{i=1}^n b_i$求余下所有序列的权值和。

题解

设每个人的传球数为 c_i，不难发现 $\min(c_i)\gt 0$ 时所有 c_i 减一最终得到的序列 B 不变。

于是不妨假设 $\min(c_i)=0$，不难发现在此基础上不同的序列 C 对应不同的序列 B。

利用容斥，答案就等于 c_i 不限制时产生的所有序列 B 的权值和减去 c_i 强制都大于 0 时产生的所有序列 B 的权值和。

接下来考虑如何计算 c_i 无限制时的答案。首先不考虑环的情况，设 $\text{dp}(i,j)$ 表示第 $c_i=j$ 时只考虑前 i 个人产生的序列的权值和，于是有

$$ \text{dp}(i,j)=\sum_{k=0}^{c_{i-1}}\text{dp}(i-1,k)(c_i+k-j) $$

考虑加速 dp，设 $s_1(i)=\sum_{j=0}^{c_i}\text{dp}(i,j),s_2(i)=\sum_{j=0}^{c_i}j\times \text{dp}(i,j)$，于是有

$$ \text{dp}(i,j)=(c_i-j)s_1(i-1)+s_2(i-1)\\
s_1(i)=\sum_{j=0}^{c_i}\text{dp}(i,j)=\sum_{j=0}^{c_i}((c_i-j)s_1(i-1)+s_2(i-1))=(c_is_1(i-1)+s_2(i-1))\sum_{j=0}^{c_i}1-s_1(i-1)\sum_{j=0}^{c_i}j\\
s_2(i)=\sum_{j=0}^{c_i}j\times \text{dp}(i,j)=\sum_{j=0}^{c_i}j((c_i-j)s_1(i-1)+s_2(i-1))=(c_is_1(i-1)+s_2(i-1))\sum_{j=0}^{c_i}j-s_1(i-1)\sum_{j=0}^{c_i}j^2 $$

于是上式可以 $O(n)$ 递推。最后考虑环的处理，不难发现 b_i 的贡献可以分解为本来属于自己的球和来自别人的球。

于是答案可以分解 $\prod_{i=1}^n b_i=(b_1-c_n)\prod_{i=2}^n b_i+c_n\prod_{i=2}^n b_i$。

对于 $(b_1-c_n)\prod_{i=2}^n b_i$，由于只考虑第一个人自己的球，于是有 $\text{dp}(1,i)=a_1-i$，最后贡献为 $\sum_{i=0}^{a_n}\text{dp}(n,i)=s_1(n)$。

对于 $c_n\prod_{i=2}^n b_i$，由于只考虑第 n 个人传给第一个人的球，于是有 $\text{dp}(1,i)=1$，最后贡献为 $\sum_{i=0}^{a_n}i\times \text{dp}(n,i)=s_2(n)$。

关于 c_i 强制都大于 0 的情况，无非是部分公式的下标从 1 开始，计算过程与上述情况类似。

```
const int MAXN=1e5+5,mod=998244353;
int quick_pow(int n,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*n%mod;
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
int inv2,inv6,a[MAXN],dp[2][MAXN];
int pre1(int n){
    return 1LL*n*(n+1)%mod*inv2%mod;
}
int pre2(int n){
```

```
    return 1LL*n*(n+1)%mod*(2*n+1)%mod*inv6%mod;
}
void calc(int n,int s1,int s2,bool zero){
    dp[0][1]=s1,dp[1][1]=s2;
    _rep(i,2,n){
dp[0][i]=((1LL*a[i]*dp[0][i-1]+dp[1][i-1])%mod*(a[i]+zero)-1LL*dp[0][i-1]*p
re1(a[i]))%mod;
dp[1][i]=((1LL*a[i]*dp[0][i-1]+dp[1][i-1])%mod*pre1(a[i])-1LL*dp[0][i-1]*pr
e2(a[i]))%mod;
    }
}
int solve(int n,bool zero){
    calc(n,a[1]+zero,pre1(a[1]),zero);
    int t=dp[1][n];
    calc(n,(1LL*(a[1]+zero)*a[1]-pre1(a[1]))%mod,(1LL*pre1(a[1])*a[1]-
pre2(a[1]))%mod,zero);
    return (t+dp[0][n])%mod;
}
int main(){
    int n=read_int();
    _rep(i,1,n)
    a[i]=read_int();
    inv2=quick_pow(2,mod-2);
    inv6=quick_pow(6,mod-2);
    int ans=(solve(n,true)-solve(n,false))%mod;
    ans=(ans+mod)%mod;
    enter(ans);
    return 0;
}
```