比赛链接

补题情况

题目	蒋贤蒙	王赵安	王智彪
A	0	0	0
B	2	0	2
C	2	0	1
D	2	0	0
E	2	0	1
F	2	0	2
G	2	0	2
H	0	0	2
I	2	0	2
J	2	0	2

题解

D. Rebuild Tree

题意

给定一棵树，要求删除 k 条边再补上 k 条边，使得新图仍然是一棵树，问方案数。

题解

首先删除 k 条边得到 $k+1$ 个连通块，设第 i 个连通块有 s_i 个点。

固定 $(s_1,s_2\cdots s_{k+1})$将每个连通块缩点后构造 prufer 序列。设第 i 个连通块在新图中得度数为 d_i知每个生成树的贡献为

$$ \prod_{i=1}^{k+1}s_i^{d_i} $$

设 p_i 表示第 i 个连通块在 prufer 序列中出现的次数，根据 prufer 序列性质，知 $d_i=p_i+1$于是有

$$ \prod_{i=1}^{k+1}s_i^{d_i}=
\left(\prod_{i=1}^{k+1}s_i\right)\left(\prod_{i=1}^{k+1}s_i^{p_i}\right) $$

考虑 $\prod_{i=1}^{k+1}s_i^{p_i}$ 在 prufer 序列中的实际意义。

不难发现设 prufer 序列的权值为每个元素 a_i 按权值 s_{a_i} 相乘之积，则所有 $\prod_{i=1}^{k+1}s_i^{p_i}$ 之和等价于所有 prufer 序列的权值之和。

不难发现，所有 prufer 序列的权值之和(固定其他位置，枚举单个位置的变化)等于

$$ \left(\sum_{i=1}^{k+1} s_i\right)^{k-1}=n^{k-1} $$

于是对固定 $(s_1,s_2\cdots s_{k+1})$ 总贡献为

$$
\left(\prod_{i=1}^{k+1}s_i\right)\sum_{Tree(k+1)}\left(\prod_{i=1}^{k+1}s_i^{p_i}\right)=\left(\prod_{i=1}^{k+1}s_i\right)\left(\sum_{i=1}^{k+1}
s_i\right)^{k-1}=\left(\prod_{i=1}^{k+1}s_i\right)n^{k-1} $$

接下来考虑 $(s_1,s_2\cdots s_{k+1})$ 不固定时产生的总贡献。

不难发现 $\prod_{i=1}^{k+1}s_i$ 等于连通块的大小之积，这可以转化为在每个连通块中选一个点的方案数。

于是设 $\text{dp}(u,k,0/1)$ 表示以 u 为子树已经割 k 条边，$0/1$ 表示 u 所在的连通块是否已经选择代表节点的方案数。

进行树形 dp 转移，枚举是否割断 u 与 $v\in \text{son}(u)$ 的连边的贡献，注意如果 v 所在的连通块没有代表节点，则 $u\to v$ 一定不可割。

时间复杂度 $O(nk)$

```
const int MAXN=5e4+5,MAXK=105,mod=998244353;
struct Edge{
    int to,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
int quick_pow(int a,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*a%mod;
        a=1LL*a*a%mod;
        k>>=1;
    }
    return ans;
}
void add(int &x,LL y){
    x=(x+y)%mod;
}
int sz[MAXN],dp[MAXN][MAXK][2],temp[MAXK][2];
void dfs(int u,int fa){
    sz[u]=1;
    dp[u][0][0]=dp[u][0][1]=1;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs(v,u);
```

```
        _for(j,0,min(sz[u]+sz[v],MAXK))
        temp[j][0]=temp[j][1]=0;
        _for(j,0,min(sz[u],MAXK))_for(k,0,min(sz[v],MAXK)){
            if(j+k<MAXK){
                add(temp[j+k][0],1LL*dp[u][j][0]*dp[v][k][0]);
add(temp[j+k][1],1LL*dp[u][j][0]*dp[v][k][1]+1LL*dp[u][j][1]*dp[v][k][0]);
            }
            if(j+k+1<MAXK){
                add(temp[j+k+1][0],1LL*dp[u][j][0]*dp[v][k][1]);
                add(temp[j+k+1][1],1LL*dp[u][j][1]*dp[v][k][1]);
            }
        }
        _for(j,0,min(sz[u]+sz[v],MAXK)){
            dp[u][j][0]=temp[j][0];
            dp[u][j][1]=temp[j][1];
        }
        sz[u]+=sz[v];
    }
}
int main(){
    int n=read_int(),k=read_int();
    _for(i,1,n){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
    }
    dfs(1,0);
    enter(1LL*quick_pow(n,k-1)*dp[1][k][1]%mod);
    return 0;
}
```

G. Product

题意

给定 n,k,D求

$$ \sum_{\sum a_i=D,a_i\ge 0}\frac {D!}{\prod_{i=1}^n (a_i+k)} $$

题解

不妨设 $b_i=a_i+k$

$$ \sum_{\sum a_i=D,a_i\ge 0}\frac {D!}{\prod_{i=1}^n (a_i+k)}=\frac {D!}{(D+nk)!}\sum_{\sum b_i=D+nk,b_i\ge k}\frac {(D+nk)!}{\prod_{i=1}^n b_i} $$

不难发现，式子右半部分的组合意义是由元素 $1\sim n$ 构成的长度为 $D+nk$ 的排列个数，其中每个元素出现次数不小于 k

不妨考虑容斥，设 $\text{dp}(i,j)$ 表示由元素 $1\sim i$ 构成的长度为 j 的排列个数，其中每个元素出现次数小于 k不难得到状态转移

$$ \text{dp}(i,j)=\sum_{v=0}^{\min(j,k)}{j\choose v}\text{dp}(i-1,j-v) $$

同时有公式 n 种元素出现次数无限制时构成的长度为 k 的排列个数为 n^k于是有

$$ \sum_{\sum b_i=D+nk,b_i\ge k}\frac {(D+nk)!}{\prod_{i=1}^n b_i}=\sum_{i=0}^n (-1)^n\sum_{j=0}^{ik}{D+nk\choose j}\text{dp}(i,j)(n-i)^{D+nk-j} $$

总时间复杂度 $O\left(n^2k^2\right)$

```
const int MAXN=55,MAXV=MAXN*MAXN,mod=998244353;
int quick_pow(int a,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*a%mod;
        a=1LL*a*a%mod;
        k>>=1;
    }
    return ans;
}
int frac[MAXV],invf[MAXV],down[MAXV];
int C(int n,int m){
    return 1LL*frac[n]*invf[m]%mod*invf[n-m]%mod;
}
int C2(int n){
    return 1LL*down[n]*invf[n]%mod;
}
int dp[MAXN][MAXV];
int main(){
    int n=read_int(),k=read_int(),D=read_int();
    //pre
    frac[0]=1;
    _for(i,1,MAXV)frac[i]=1LL*frac[i-1]*i%mod;
    invf[MAXV-1]=quick_pow(frac[MAXV-1],mod-2);
    for(int i=MAXV-1;i;i--)
    invf[i-1]=1LL*invf[i]*i%mod;
    down[0]=1;
    _for(i,0,n*k)
    down[i+1]=1LL*down[i]*(D+n*k-i)%mod;
    //main
    LL ans=quick_pow(n,D+n*k);
    dp[0][0]=1;
    _rep(i,1,n){
        _for(j,0,i*k){
            _for(v,0,k){
                if(j>=v)
                dp[i][j]=(dp[i][j]+1LL*dp[i-1][j-v]*C(j,v))%mod;
            }
```

```
            int d=1LL*dp[i][j]*C(n,i)%mod*C2(j)%mod*quick_pow(n-i,D+n*k-
j)%mod;
            if(i&1)
            ans=(ans+mod-d)%mod;
            else
            ans=(ans+d)%mod;
        }
    }
    int t=1;
    _rep(i,D+1,D+n*k)
    t=1LL*t*i%mod;
    ans=ans*quick_pow(t,mod-2)%mod;
    enter(ans);
    return 0;
}
```