比赛链接

补题情况

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

题解

C. Fuzzy Graph

题意

给定一个连通图，要求将图上每个点染成 RGB 三种颜色。对应边 (u,v)如果 u,v 同色，则该边染成 u,v 的颜色，否则该边染成黑色。

要求构造一种染色方案，使得删除图中所有非黑边仍然可以使图连通，同时满足如下两个条件之一：

1. RGB 三种颜色的点一样多，数据保证 $3\mid n$
2. X 是点集中出现次数最多的颜色(允许有其他颜色出现次数和他相同)，且不存在边的颜色为 X

题解

首先跑一遍 dfs 树，然后用 RG 两种颜色进行二染色，这样所有树边都是黑色边，图保证连通。

接下来，设 c_0 表示 R 颜色在点集中出现的次数，设 c_1 表示 G 颜色在点集中出现的次数。

若 $c_0\le \frac n3$ 或者 $c_1\le \frac n3$则考虑构造第二个条件：将所有叶子结点染成颜色 B

事实上，由于叶子结点在 dfs 树上只有返祖边，所以所有叶子节点之间一定没有连边，即不存在颜色为 B 的边。

另一方面，不妨设 $c_0\le \frac n3$则所有颜色为 G 的非叶子结点至少是一个颜色为 R 的结点的父亲，所以所有颜色为 G 的非叶子结点个数不超过 $\frac n3$

由于颜色为 R 的非叶子结点个数不超过颜色为 R 的结点总个数 c_0所有颜色为 R 的非叶子结点个数也不超过 $\frac n3$

所以叶子结点总个数一定不小于 $\frac n3$且将叶子结点染成 B 后 B 出现次数一定最多。

若 $c_0\ge \frac n3$ 且 $c_1\ge \frac n3$则考虑构造第一个条件：从深到浅贪心染色，如果当前结点颜色出现个数大于 $\frac n3$ 且相邻结点颜色不为 B则染成 B

事实上，设 $c_0=\frac n3+d_0,c_1=\frac n3+d_1$根据上述构造，易知 G 被染色成 B 的个数不超过 d_1

而由于是从深到浅染色，所以每个从 G 被染色成 B 的结点只会 ban 他的父结点，不考虑 ban 儿子结点。

所有 R 结点的可选染色位置最多被 ban d_1 个位置，于是要从余下 $\frac n3+d_0-d_1$ 个位置中选出 d_0 个位置，这一定是可行的。

同理，所有 G 结点一定也可以选出 d_1 个位置染成 B

总时间复杂度 $O(n+m)$

```
const int MAXN=3e5+5,MAXM=5e5+5;
const char s[]="RGB";
struct Edge{
    int to,next;
}edge[MAXM<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
int ans[MAXN],fa[MAXN],d[MAXN],cnt[2];
bool fib[MAXN];
vector<int> node[MAXN],leaf;
void dfs(int u,int f){
    ans[u]=ans[f]^1;
    cnt[ans[u]]++;
    d[u]=d[f]+1;
    fa[u]=f;
    node[d[u]].push_back(u);
    bool flag=false;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==f||d[v])continue;
        flag=true;
        dfs(v,u);
    }
    if(!flag)
    leaf.push_back(u);
}
void solve(){
    int n=read_int(),m=read_int();
    _rep(i,1,n){
        head[i]=0;
```

```
        d[i]=0;
        fib[i]=false;
        node[i].clear();
    }
    cnt[0]=cnt[1]=0;
    leaf.clear();
    edge_cnt=0;
    while(m--){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
    }
    dfs(1,0);
    if(cnt[0]<=n/3||cnt[1]<=n/3){
        for(int v:leaf)
        ans[v]=2;
    }
    else{
        for(int i=n-1;i;i--){
            for(int v:node[i]){
                if(fib[v]||cnt[ans[v]]==n/3)continue;
                cnt[ans[v]]--;
                ans[v]=2;
                fib[fa[v]]=true;
            }
        }
    }
    _rep(i,1,n)
    putchar(s[ans[i]]);
    putchar('\n');
}
int main(){
    int T=read_int();
    while(T--)
    solve();
    return 0;
}
```

H.Scholomance Academy

注意到 $F(n)$ 是一个积性函数，因为对于两堆不同的质因子放进去，因为欧拉函数是积性函数，我都可以分成两堆，一堆装第一类，一堆装第二类，然后发现和单独相乘是一一对应的关系，必然相等。

于是我们单独对每个质因子拆开计算。

我们考虑生成函数 $G_{p}(x)=F(p^{0})+F(p^{1})x^{1}+\ldots$ 然后所有质因子的生成函数乘起来的 n 次项就是 $G(N)$ 的结果了。

这东西等于 $(\phi (p^{0})+\phi (p^{1})x^{1}+\ldots)^{m}$

化简得到 $(\frac {1-x} {1-px})^{m}$ ，然后所有的质因子乘起来，最后分子分母都是最高次项为 mt 次方的多项式，设除的结果为 $h(x)$ ，我们要求 $h(x)$ 的第 n 次项，设分子为 $f(x)$ ，分母为 $g(x)$ 。

然后这东西是可以常系数齐次线性递推搞的：对于高于 mt 次的系数，$[x^{n}]f(x)$ 必然是 0 ，然后有 $[x^{n}]f(x)=0=\sum_{i=0}^{mt}[x^{n-i}]h(x)\times[x^{i}]g(x)$ ，然后我们就有 $[x^{n}]h(x)\times[x^{0}]g(x)=-\sum_{i=1}^{mt}[x^{n-i}]h(x)\times[x^{i}]g(x)$ ，这玩意求逆元化简之后，就变成了一个常系数齐次线性递推的板子。

这里有两个需要注意的地方， n 比较小的时候，是要直接输出多项式逆元乘法的第 n 项，然后递推的时候递推的系数是 $[x^{1}]g(x)$ 到 $[x^{mt}]g(x)$ 然后因为 $[x^{mt}]f(x)\neq0$ ，所以这个 n 要从 $mt+1$ 开始取，也就是说要我们初始项是 $[x^{1}]h(x)$ 到 $[x^{mt}]h(x)$ 然后就需要向前挪一项，然后求 $n-1$ 次的系数。

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N=2000006,GG=3,GI=332748118,mod=998244353;

inline void read(int &X) {
    X = 0;
    int w=0;
    char ch=0;
    while(!isdigit(ch)) {
        w|=ch=='-';
        ch=getchar();
    }
    while(isdigit(ch)) X=(X<<3)+(X<<1)+(ch^48),ch=getchar();
    if (w) X = -X;
}
namespace my_f {
    int qpow(int a,int b,int m=mod) {
        int r=1;
        while(b) {
            if (b&1) r=r*a%m;
            a=a*a%m,b>>=1;
        }
        return r;
    }
    int pgg[50],pgi[50];
    void init() {
        for(int i=0; i<=48; i++) {
            pgg[i]=qpow(GG,(mod-1)>>i);
            pgi[i]=qpow(GI,(mod-1)>>i);
        }
    }
#define ad(x,y) ((x)+(y)>mod?(x)+(y)-mod:(x)+(y))
#define dc(x,y) ((x)-(y)<0?(x)-(y)+mod:(x)-(y))
    namespace poly {
        int w[N],r[N];
        void NTT(int f[N],int lim,int type) {
```

```
            for(int i=0; i<lim; i++) if (i<r[i]) swap(f[i],f[r[i]]);
            for(int mid=1,pp=1; mid<lim; mid<<=1,++pp) {
                int Wn=(type==1?pgg[pp]:pgi[pp]);
                w[0]=1;
                for(int i=1; i<mid; i++) w[i]=w[i-1]*Wn%mod;
                for(int i=0; i<lim; i+=(mid<<1)) {
                    for(int j=0; j<mid; ++j) {
                        int y=f[i|mid|j]*w[j]%mod;
                        f[i|mid|j]=dc(f[i|j],y);
                        f[i|j]=ad(f[i|j],y);
                    }
                }
            }
            if (type==-1) {
                int ivv=qpow(lim,mod-2);
                for(int i=0; i<lim; i++) f[i]=f[i]*ivv%mod;
            }
        }
        void init(int n) {
            r[0]=0;
            for(int i=1; i<n; i++)r[i]=((r[i>>1]>>1)|((i&1)*(n>>1)));
        }
        int pw[2][N],rev[N];
        void pre() {
            for(int i=1; i<N;
i++)pw[0][i]=qpow(GG,(mod-1)/(i<<1)),pw[1][i]=qpow(GG,mod-1-
(mod-1)/(i<<1));
        }
        void vec_NTT(vector<int> &a,int n,int o) {
            for(int i=0; i<n; i++)if(i<r[i])swap(a[i],a[r[i]]);
            for(int i=1; i<n; i<<=1) {
                int wn;
                if(o==1)wn=pw[0][i];
                else wn=pw[1][i];
                for(int j=0; j<n; j+=(i<<1)) {
                    int w=1;
                    for(int k=0; k<i; k++) {
                        int t=w*a[i+j+k]%mod;
                        w=w*wn%mod;
                        a[i+j+k]=(a[j+k]-t+mod)%mod;
                        a[j+k]=(a[j+k]+t)%mod;
                    }
                }
            }
            if(o==-1) {
                int INV=qpow(n,mod-2);
                for(int i=0; i<=n; i++)a[i]=a[i]*INV%mod;
            }
        }
        int pool[10][N];
        inline void Inv(int f[N],int g[N],int n) { //求逆, pool 0~2
```

```
            int *iv=pool[0],*a=pool[1],*b=pool[2];
            for(int i=0; i<=4*n; i++) iv[i]=a[i]=b[i]=0;
            iv[0]=qpow(f[0],mod-2);

            int len=1,lim=1;
            for(len=1; len<=(n<<1); len<<=1) {
                lim=len<<1;

                for(int i=0; i<=4*len; i++) a[i]=b[i]=0;
                for(int i=0; i<len; i++) a[i]=f[i],b[i]=iv[i];
                for(int i=0; i<lim; i++) r[i]=((r[i>>1]>>1)|((i&1)?len:0));
                NTT(a,lim,1);
                NTT(b,lim,1);
                for(int i=0; i<lim; i++) a[i]=(2*b[i]-
a[i]*b[i]%mod*b[i]%mod+mod)%mod;
                NTT(a,lim,-1);
                for(int i=0; i<len; i++) iv[i]=a[i];
            }
            for(int i=0; i<n; i++) g[i]=iv[i];
            for(int i=n; i<lim; i++) g[i]=0;
        }
        vector<int> vec_mul(vector<int> a,vector<int> b) {
            int len=a.size()+b.size()-1,lim=1;
            while(lim<=len) lim<<=1;
            init(lim);
            a.resize(lim);
            vec_NTT(a,lim,1);
            b.resize(lim);
            vec_NTT(b,lim,1);
            for(int i=0; i<lim; i++)a[i]=a[i]*b[i]%mod;
            vec_NTT(a,lim,-1);
            a.resize(len);
            return a;
        }
        inline void mul(int f[N],int g[N],int n,bool flag=1) // *=, pool
3~4
        // flag: 是否对n取膜
        {
            int len=1,lim=1;
            while(len<=(n<<1)) len=lim,lim<<=1;
            int *a=pool[3],*b=pool[4];
            for(int i=0; i<lim; i++) a[i]=b[i]=0;
            for(int i=0; i<n; i++) a[i]=f[i],b[i]=g[i];
            for(int i=0; i<lim; i++) r[i]=((r[i>>1]>>1)|((i&1)?len:0));
            NTT(a,lim,1);
            NTT(b,lim,1);
            for(int i=0; i<lim; i++) a[i]=a[i]*b[i]%mod;
            NTT(a,lim,-1);
            for(int i=0; i<2*n; i++) f[i]=a[i];
            for(int i=2*n; i<=lim; i++) f[i]=0;
```

```
            if (flag) for(int i=n; i<2*n; i++) f[i]=0;
        }
        void Mod(int f1[N],int f2[N],int n,int m,int R[N]) //多项式取模, pool 5~6
        //这里只保留了余数,商开到 pool 里而不是传参数修改了
        {
            int *a=pool[5],*b=pool[6],*Q=pool[7];
            for(int i=0; i<=4*n; i++) a[i]=b[i]=0;
            for(int i=0; i<n; i++) a[i]=f1[n-1-i];
            for(int i=n-m+1; i<n; i++) a[i]=0; // a=f1_r%(x^(n-m+1))
            for(int i=0; i<m; i++) b[i]=f2[m-1-i];
            for(int i=n-m+1; i<m; i++) b[i]=0;
            Inv(b,b,n-m+1);
            mul(a,b,n-m+1);
            for(int i=0; i<=n-m; i++) Q[i]=a[n-m-i];

            for(int i=0; i<=4*n; i++) a[i]=b[i]=0;
            for(int i=0; i<n; i++) a[i]=f1[i];
            for(int i=0; i<m; i++) b[i]=f2[i];
            int len=1,lim=1;
            while(len<=n) len=lim,lim<<=1;
            for(int i=0; i<lim; i++) r[i]=((r[i>>1]>>1)|((i&1)?len:0));
            NTT(b,lim,1);
            NTT(Q,lim,1);
            for(int i=0; i<lim; i++) b[i]=b[i]*Q[i]%mod;
            NTT(b,lim,-1);
            NTT(Q,lim,-1);
            for(int i=0; i<m-1; i++) R[i]=(a[i]-b[i]%mod+mod)%mod;
            for(int i=m-1; i<lim; i++) R[i]=0;
        }
        void PowMod(int f[N],int p,int g[N],int n,int h[N]) { // f^p%g, g有n项,保存在h
            int *res=pool[8];
            for(int i=0; i<=8*n; i++) res[i]=0;
            res[0]=1; // res=1
            while(p) {
                if (p&1) {
                    mul(res,f,n,0); // res*=f
                    Mod(res,g,2*n,n,res); // res%=g
                }
                mul(f,f,n,0); // f=f*f
                Mod(f,g,2*n,n,f); // f%=g
                p>>=1;
            }
            for(int i=0; i<n-1; i++) h[i]=res[i];
            for(int i=n; i<=8*n; i++) h[i]=0;
        }
    }
    int n,T,m;
    int
a[N],t[N],p[N],F[N],tmp[N],tmp1[N],tmp2[N],tmp3[N],fm[N],fm1[N],jc[N],inv[N
```

```
];
    vector<int> work(int l,int r) {
        int mid=l+r>>1;
        if(l==r) {
            vector<int> ans(2);
            ans[0]=1,ans[1]=mod-p[l];
            return ans;
        }
        return poly::vec_mul(work(l,mid),work(mid+1,r));
    }

    void Input() {
        init();
        poly::pre();
        read(n);
        read(T);
        read(m);
        for(int i=1; i<=T; i++) read(p[i]);
        fm[0]=1;
        jc[0]=jc[1]=1;
        for(int i=2; i<=m*T+1; i++) {
            jc[i]=jc[i-1]*i%mod;
        }
        inv[0]=inv[1] = 1;
        for(int i=2; i<=m*T+1; ++i) {
            inv[i]=(mod-mod/i)*inv[mod%i]%mod;
        }
        for(int i=2; i<=m*T+1; ++i) {
            inv[i]=inv[i]*inv[i-1]%mod;
        }
        vector<int> mkbk=work(1,T);
        vector<int> fm2(1);
        fm2[0]=1;
        for(int i=1; i<=m; i++) {
            fm2=poly::vec_mul(fm2,mkbk);
        }
        for(int i=0,flag=1; i<=m*T; i++,flag=-flag) {
            tmp3[i]=flag*(jc[m*T]*inv[i]%mod*inv[m*T-i]%mod)%mod;
            if(tmp3[i]<0) tmp3[i]+=mod;
        }

        for(int i=0; i<fm2.size(); i++) {
            fm[i]=fm2[i];
        }
        long long ttmp=qpow(fm[0],mod-2);
        for(int i=1; i<=m*T; i++) {
            fm1[i]=(mod-fm[i])*ttmp%mod;
        }
        poly::Inv(fm,tmp2,m*T+1);
        poly::mul(tmp2,tmp3,m*T+1,1);
```

```
        if(n<=m*T) {
            printf("%lld\n",tmp2[n]);
            exit(0);
        }
        for(int i=m*T+1; i<=2*m*T; i++) tmp2[i]=0;
        for(int i=0;i<=m*T;i++) {
            tmp2[i]=tmp2[i+1];
        }
    }
    int f[N],g[N],c[N];
    void Sakuya() {
        f[1]=1;
        for(int i=1; i<=m*T; i++) g[m*T-i]=(mod-fm1[i]);
        g[m*T]=1;
        poly::PowMod(f,n-1,g,m*T+1,c);

        int ans=0;
        for(int i=0; i<m*T; i++) ans=(ans+tmp2[i]*c[i]%mod)%mod;
        printf("%lld\n",ans);
    }

}
#undef int
using namespace my_f;
int main() {
    Input();
    Sakuya();
    return 0;
}
```

J. Tree

题意

给定一棵树和两个人的初始位置 s,t，两个人在树上轮流移动，两个人走过的点都不能再走且两个人不能在同一个点。

每个人的分数为自己无法移动时走过的点数，两个人都希望最大化自己和另一个人的分数差，问最终先手的分数减去后手分数的值。

题解

首先以先手为根建树，树形 dp 求出每个点 u 不经过 s,t 链上的点能到达的最远距离 $a(u)$，设链上的点为 $v_1,v_2\cdots v_k$，其中 $v_1=s,v_k=t$。

定义 $\text{solve}(l,r,0/1)$ 表示第一个人位于 v_l 第二个人位于 v_r，$0/1$ 表示轮到先手/后手行动的答案。

对 $\text{solve}(l,r,0)$ 先手如果选择进入子树，此时答案为 $a(v_l)+l-1-\max_{l\lt x\le r}(a(v_x)+k-x)$

如果选择不进入子树，则答案为 $\text{solve}(l+1,r,1)$ 所以两种情况取 $\max$ 即可，后手情况类似。

上述转移可以使用 ST 表，时间复杂度 $O(n\log n)$

```
const int MAXN=5e5+5,MAXM=20;
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 p[MAXN];
multiset<int> dp[MAXN];
void dfs(int u,int fa){
    dp[u].insert(1);
    p[u]=fa;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs(v,u);
        dp[u].insert(*dp[v].rbegin()+1);
    }
}
struct ST{
    int d[MAXN][MAXM],lg2[MAXN];
    void build(int *a,int n){
        lg2[1]=0;
        _rep(i,2,n)
        lg2[i]=lg2[i>>1]+1;
        _for(i,0,n)
        d[i][0]=a[i];
        for(int j=1;(1<<j)<=n;j++){
            _for(i,0,n+1-(1<<j))
            d[i][j]=max(d[i][j-1],d[i+(1<<(j-1))][j-1]);
        }
    }
    int query(int lef,int rig){
        int len=rig-lef+1;
        return max(d[lef][lg2[len]],d[rig-(1<<lg2[len])+1][lg2[len]]);
    }
}st1,st2;
int mv1[MAXN],mv2[MAXN];
int solve(int l,int r,int pos){
    if(l+1==r)
    return mv1[l]-mv2[r];
    if(pos==0)
```

```
    return max(mv1[l]-st2.query(l+1,r),solve(l+1,r,!pos));
    else
    return min(st1.query(l,r-1)-mv2[r],solve(l,r-1,!pos));
}
int main(){
    int n=read_int(),s=read_int(),t=read_int();
    _for(i,1,n){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
    }
    dfs(s,0);
    vector<int> vec;
    int pos=t;
    while(pos){
        vec.push_back(pos);
        pos=p[pos];
    }
    reverse(vec.begin(),vec.end());
    _for(i,0,vec.size()){
        int v=vec[i];
        if(i!=vec.size()-1){
            int v2=vec[i+1];
            dp[v].erase(dp[v].find(*dp[v2].rbegin()+1));
        }
    }
    _for(i,0,vec.size()){
        mv1[i]=*dp[vec[i]].rbegin()+i;
        mv2[i]=*dp[vec[i]].rbegin()+vec.size()-i-1;
    }
    st1.build(mv1,vec.size());
    st2.build(mv2,vec.size());
    enter(solve(0,vec.size()-1,0));
    return 0;
}
```