A - 字符串

观察一下删 $[i,i+k],[j,j+k]$ 等价的条件 $(i<j)$ ，发现要求 $s[i+k+1,j+k]=s[i,j-1]$ 。如果这个东西满足，可以发现 $[i,i+k],[i+1,i+1+k],\dots,[j-1,j-1+k],[j,j+k]$ 都是等价的。

于是对于固定的 k ，能造出的等价类个数其实就是 $s_i\neq s_{i+k+1}$ 的位置个数再加上 1 。

发现答案就是 $s_i\neq s_j$ 的 (i,j) 个数加上 n 。令 t_i 是等于 i 的 s 个数，答案即 $n+\frac{\sum t_i(n-t_i)}{2}$ 。

```
#include<bits/stdc++.h>
using namespace std;
char c[1000100];
int t[26],n;
int main(){
    scanf("%d%s",&n,c+1);
    for(int i=1;i<=n;i++)t[c[i]-'a']++;
    long long ans=0;
    for(int i=0;i<26;i++)ans+=1ll*t[i]*(n-t[i]);
    ans/=2,ans+=n;printf("%lld",ans);
    return 0;
}
```

B - 选举

考虑容斥，我们会钦定一些人投给自家人。

先想一下钦定一部分人后的方案数怎么算，令 x_i 表示一个人接收到的钦定票数，发现方案即为 $\frac{(n-\sum x_i)!}{\prod(c_i-x_i)!}$ 。

那来思考，如果确定了序列 x ，怎么算有多少种钦定方式会得到 x 呢？发现我们只需要对每个家庭分别考虑，答案即 $\prod\limits_{i=1}^{n}\frac{a_i!}{(a_i-\sum\limits_{t_j=i}x_j)!\prod\limits_{t_j=i}x_j!}$ ，其中令 a_i 表示家庭 i 的人数。

那么一个序列 x 给答案带来的贡献即为：

$(-1)^{\sum x_i}\frac{(n-\sum x_i)!}{\prod(c_i-x_i)!}\prod\limits_{i=1}^{n}\frac{a_i!}{(a_i-\sum\limits_{t_j=i}x_j)!\prod\limits_{t_j=i}x_j!}$ ，其中 $(-1)^{x_i}$ 是容斥系数。

整理一下式子，把只和一个人相关的部分整理到一起，即 $\prod a_i!*\prod\frac{(-1)^{x_i}}{(c_i-x_i)!x_i!}*(n-\sum x)!*\prod\limits_{i=1}^{n}\frac{1}{(a_i-\sum\limits_{t_j=i}x_j)!}$ 。

现在对每个家庭分别求出 f_i 表示满足 $\sum x=i$ 的方案的 $\prod \frac{(-1)^{x_i}}{(c_i-x_i)!x_i!}$ 之和，这是容易的，可以直接 dp 出来。算完了之后把 f_i 乘上 $\frac{1}{(a-i)!}$ 。

然后开始第二层 dp ，我们只关心 $\sum x$ ，就设 $dp_{i,j}$ 是考虑了前 i 个家庭，当前 $\sum x=j$ 的所有方案的权值和，转移也是容易的。 答案即 $\prod a_i!\sum dp_{n,i}(n-i)!$ 。

复杂度 $O(n^2)$ 。

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,c[5010],t[5010],jc[5010],ij[5010];
const int mod=998244353;
int qpow(int a,int b){int c=1;for(;b;b>>=1){if(b&1)c=1ll*a*c%mod;a=1ll*a*a%mod;}return c;}
#define pb push_back
vector<int>g[5010];
int dp[5010],f[5010],f2[5010];
int main(){
    scanf("%d",&n);
    jc[0]=ij[0]=1;
    for(int i=1;i<=n;i++)scanf("%d",&c[i]),jc[i]=1ll*i*jc[i-1]%mod,ij[i]=qpow(jc[i],mod-2);
    for(int i=1;i<=n;i++)scanf("%d",&t[i]),g[t[i]].pb(c[i]);
    int J=1,S=0;
    dp[0]=1;
    for(int i=1;i<=n;i++)if(!g[i].empty()){
        memset(f,0,sizeof(f));
        int A=g[i].size();
        J=1ll*J*jc[A]%mod;
        f[0]=1;int C=0;
        for(int x:g[i]){
            memset(f2,0,sizeof(f2));
            for(int j=0;j<=C;j++)for(int k=0;k<=x;k++)
            (f2[j+k]+=((k&1)?(mod-1ll):1ll)*f[j]%mod*ij[k]%mod*ij[x-k]%mod)%=mod;
            C+=x;
            for(int j=0;j<=C;j++)f[j]=f2[j];
        }
        for(int j=0;j<=A;j++)f[j]=1ll*f[j]*ij[A-j]%mod;
        memset(f2,0,sizeof(f2));
        for(int j=0;j<=S;j++)for(int k=0;k<=A;k++)
        (f2[j+k]+=1ll*dp[j]*f[k]%mod)%=mod;
        S+=A;
        for(int j=0;j<=S;j++)dp[j]=f2[j];
    }
    int ans=0;
    for(int i=0;i<=n;i++)(ans+=1ll*jc[n-i]*dp[i]%mod)%=mod;
    printf("%lld\n",1ll*ans*J%mod);
    return 0;
}
```

bonus: $O(n\log^2 n)$?

C - 替换

考虑二分答案 mid 。

先想一想暴力怎么做。注意到每个位置的操作其实是独立的：我们可以看成先对 1 进行不超过 mid 次操作，再对 2 进行不超过 mid 次操作，这样做下去。

从左往右依次确定 a_i 最终等于多少。肯定希望它尽量小，但也需要 $\geq a_{i-1}$ 的最终值。

接下来是这个题最灵魂的一步：令 X 是 a_{i-1} 的最终值，我们只需要依次判断：从 a_i 开始能否通过至多 mid 次操作成 $X, X+1, X+2, \ldots$ 这样尝试下去，直到尝试成功就停止，得到 a_i 的最终值。

可以发现一次尝试要么成功，要么 X 会 $+1$ ，所以我们只会做至多 $n+m$ 次尝试！

问题转化成：若干次询问，每次给出 s, t ，想知道从 s 开始做多少次操作才能变成 t 。

怎么做呢，考虑把这些数看成图上的点，我们建立一个有向图，连边 (i, b_i) 。可以发现每个点出度为 1 ，即内向基环森林。现在问题变成：从 s 出发，走多少步才能走到 t 。

先考虑每个环都是自环的情况：可以忽略自环变成森林，那 s 能走到 t 等价于 t 是 s 的祖先，步数即 $d_s - d_t$ (d 是深度)。

现在不是自环，也可以先删掉环上任何一条边 (u, v) ，此时得到一颗以 u 为根的内向树。先判掉 t 是 s 的祖先的情况，和 s, t 不在一棵树的情况；接下来想走到 t 必须得先往上走到 u ， u 再跳到 v ，再尝试从 v 往上走到 t 。也就是说如果 t 是 v 祖先，答案即 $d_s + d_v - d_t + 1$ 。

可以发现预处理后单次判断是 $O(1)$ 的，本题复杂度即 $O((n+m)\log m)$ 。

```
#include<bits/stdc++.h>
using namespace std;
int n,m,a[1010010],b[1010000],fa[1010000];
vector<int>g[1001000];
int co[1010000],dk[1010000],dfn[1010000],sz[1010000],d[1010000];
bool vis[1010000];
int F(int x){if(x==fa[x])return x;return fa[x]=F(fa[x]);}
void dfs(int x){
    if(!co[x])co[x]=x;
    sz[x]=1,dfn[x]=++dfn[0];
    for(int v:g[x])co[v]=co[x],d[v]=d[x]+1,dfs(v),sz[x]+=sz[v];
}
bool zx(int a,int b){ return dfn[a]<=dfn[b]&&dfn[b]<dfn[a]+sz[a];}
int dis(int u,int v){
    if(co[u]!=co[v])return -1;
    if(zx(v,u))return d[u]-d[v];
    if(vis[v])return d[u]-d[v]+dk[co[u]]+1;
    return -1;
}
bool chk(int mi){
    int j=1;
    for(int i=1;i<=n;i++){
        while(j<=m){
            int z=dis(a[i],j);
            if(z==-1||z>mi)j++;
            else break;
        }
        if(j==m+1)return 0;
    }
    return 1;
}
void sol(){
    dfn[0]=0;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++)fa[i]=i,b[i]=0,g[i].clear(),co[i]=0,vis[i]=0;
    for(int i=1;i<=n;i++)scanf("%d",&a[i]);
    vector<int>rt;
    for(int i=1,f;i<=m;i++){
        scanf("%d",&f);
        if(F(f)!=i)fa[i]=F(f),b[i]=f,g[f].push_back(i);
        else{
            rt.push_back(i),dk[i]=0;
            int u=f;while(u)vis[u]=1,u=b[u],dk[i]++;
            dk[i]--;
        }
    }
    for(int o:rt)d[o]=0,dfs(o);
    int l=0,r=m,ans=m+1;
    while(l<=r){
        int mi=(l+r)>>1;
```

```
            if(chk(mi))ans=mi,r=mi-1;
            else l=mi+1;
        }
        if(ans==m+1)puts("-1");
        else printf("%d\n",ans);
    }
    int main(){
        sol();
        return 0;
    }
```

D - 操作

先思考如何计算 $f(s,t)$ 。我们先做一个变换，即令 s' 为一个序列满足 $s'_i = s_i \oplus [i\text{的深度是奇数}]$ ，观察每次操作后 s' 的变化，发现是直接 swap 了 s'_u 和 s'_v 。我们对 s,t 同时做变换之后再来看怎么计算操作次数。不妨以 1 为根，令 $z_i = |\sum\limits_{v\in subtree(i)} s_v - t_v|$ ，则存在方案当且仅当 $z_1 = 0$ ，且最小操作次数为 $\sum z_i$ 。这是怎么得到的呢？转化一下角度，我们把 $s_i = 1$ 的点看成是点 i 上有白棋，$t_i = 1$ 看成是点 i 上有黑棋，白棋黑棋遇上了可以抵消。虽然题意相当于只能移动白棋，但我们不妨看成：白棋，黑棋都只能向上走，这样不影响答案。于是从下往上地贪心，在当前点能抵消就尽量抵消，再把剩下的点往上移。就可以得到上面的式子了。

于是我们成功的把 $f(s,t)$ 拆成了 z 之和的形式。

现在要计算所有可能的 (a,b) 的 $f(a,b)$ 之和。首先还是对初始的 a,b 先做变换，即对于深度为奇数的点，把 a_i,b_i 异或上 1 ，如果是问号就不管。

现在令 a 中问号个数是 A， b 中问号个数是 B ，已知部分的权值和是 C ；

对于一个点 u ，令其子树内 a 问号个数为 x ，b 问号个数为 y ，已知部分权值和是 z ，那可以得到 z_u 对答案的贡献为：

$$\sum_{i\le x}\sum_{j\le y}\sum_{k\le A-x}\sum_{l\le B-y}[C+i+k-j-l=0]\binom{x}{i}\binom{y}{j}\binom{A-x}{k}\binom{B-y}{l}|z+i-j| 。$$

虽然看着很吓人，但我们可以发现两件事情：我们只关注 $i-j$ ；我们只关注 $k-l$ 。

枚举 $p=i-j,q=k-l$ 。

注意到 $\sum\limits_i \binom{x}{i}\binom{y}{i-p} = \binom{x+y}{x-p}$ ，$\sum\limits_k \binom{A-x}{k}\binom{B-y}{k-q} = \binom{A+B-x-y}{A-x-q}$ 。

于是上面的式子可以改写成：$\sum\limits_p\sum\limits_q[C+p+q=0]\binom{x+y}{x-p}\binom{A+B-x-y}{A-x-q}|z+p| = \sum\limits_p\binom{x+y}{x-p}\binom{A+B-x-y}{A-x+C+p}|z+p|$ 。

直接枚举即可单次 $O(n)$ 计算，复杂度 $O(n^2)$ ，考虑如何优化。

我们再改写一下式子，令 $p := x-p$ ，即 $\sum\limits_p\binom{x+y}{p}\binom{A+B-x-y}{A+C-p}|z+x-p|$ 。令 $f(x,y) = \sum\limits_i\binom{x}{i}\binom{X-x}{Y-i}|y-i|$ ，其中 $X=A+B, Y=A+C$ 。

接下来我们尝试 $O(1)$ 地从 $f(x,y)$ 推到 $f(x,y+1)$ 和 $f(x+1,y)$ 。

如果能做到这一点，那考虑对树重链剖分后，我们依次对每条重链计算 f 。对于一条重链，可以发现从从链顶算到链尾的过程中 x,y 的变化量是和链顶的子树大小一个级别的，这是 $O(n\log n)$ 的。（事实上直接莫队也不好卡）现在就只需要思考如何做到上述的事情了。

继续简化 $f(x,y)$ ，把绝对值拆了，即 $\sum\limits_i\binom{x}{i}\binom{X-x}{Y-i}(i-y)+2\sum\limits_{i\le y}\binom{x}{i}\binom{X-x}{Y-i}(y-i)$ 。前面一个式子可以直接 $O(1)$ 计算，我们来看后面的式子。先考虑 $\sum\limits_{i\le y}\binom{x}{i}\binom{X-x}{Y-i}i$ 这个式子，它等于 $x\sum\limits_{i\le y-1}\binom{x-1}{i}\binom{X-x}{Y-1-i}$ 。

现在就只需要研究 $g(x,y)=\sum\limits_{i\le y}\binom{x}{i}\binom{X-x}{Y-i}$ 是如何 $O(1)$ 推到 $g(x,y+1)$ 和 $g(x+1,y)$ 的。

从 $g(x,y)$ 推到 $g(x,y+1)$ 非常容易：加上 $\binom{x}{y+1}\binom{X-x}{Y-(y+1)}$ 即可。再看从 $g(x,y)$ 推到 $g(x+1,y)$ ，这可以从组合意义来推：$g(x,y)$ 相当于从 X 个球中选 Y 个染色，要求前 x 个球中至多 y 个被染了。

于是 $g(x,y)-g(x+1,y)$ 等同于前 x 个球恰好被染了 y 个，且第 $x+1$ 个球被染的方案数。这就是 $\binom{x}{y}\binom{X-1-x}{Y-1-y}$ ！需要注意在一些边界情况下这个组合意义是不合法的，需要特殊计算。

总复杂度 $O(n\log n)$ 。

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e5+10;
int n,fa[N];
vector<int>g[N];
#define pb push_back
char c1[N],c2[N];
const int mod=1e9+7;
int qpow(int a,int b){
    int c=1;
    for(;b;b>>=1){
        if(b&1)c=1ll*a*c%mod;
        a=1ll*a*a%mod;
    }
    return c;
}
int s[N][2][3],jc[N],ij[N],sz[N],son[N];
int C(int a,int b){if(a<0||b<0||a<b)return 0;return 1ll*jc[a]*ij[b]%mod*ij[a-b]%mod;}
void doi(){jc[0]=ij[0]=1;for(int i=1;i<=200000;i++)jc[i]=1ll*i*jc[i-
1]%mod,ij[i]=qpow(jc[i],mod-2);}
void dfs(int x,int d){
    sz[x]=1;int mm=-1;
    for(int v:g[x])if(v!=fa[x]){
        fa[v]=x,dfs(v,d^1),sz[x]+=sz[v];
        if(mm<sz[v])mm=sz[v],son[x]=v;
        for(int i=0;i<2;i++)for(int j=0;j<3;j++)s[x][i][j]+=s[v][i][j];
    }
    if(c1[x]!='?')s[x][0][(c1[x]-'0')^d]++;
    else s[x][0][2]++;
    if(c2[x]!='?')s[x][1][(c2[x]-'0')^d]++;
    else s[x][1][2]++;
}
int a_[101000],b_[101000],be[100100],S;
void dfs2(int x){
    be[++S]=x;
    if(!son[x])return;dfs2(son[x]);
    for(int v:g[x])if(v!=fa[x]&&v!=son[x])dfs2(v);
}
struct qb{
    int f,X,Y;
    inline int bl(int a,int b){
        int s=0;
        for(int i=0;i<=a;i++)(s+=1ll*C(b,i)*C(Y-b,X-i)%mod)%=mod;
        return s;
    }
    int a,b;
    inline void ks(int x_,int y_){X=x_,Y=y_;a=b=0,f=C(Y,X);}
    inline int t1(){
        return 1ll*C(b,a+1)*C(Y-b,X-(a+1))%mod;
    }
```

```
    inline int t2(){
        if(a<0||Y<0||X<0||b>Y||b<-1||Y<X)return 0;
        if(b==-1)return C(Y,X);
        if(b==Y)return (X>a)?0:(mod-C(Y,X));
        return mod-1ll*C(b,a)*C(Y-1-b,X-1-a)%mod;
    }
    inline void doi(int a_,int b_){
        while(a<a_)(f+=t1())%=mod,a++;
        while(a>a_)a--,(f-=t1())%=mod;
        while(b<b_)(f+=t2())%=mod,b++;
        while(b>b_)b--,(f-=t2())%=mod;
    }
}q1,q2;
void sol(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        g[i].clear(),fa[i]=0,son[i]=0;
        for(int j=0;j<2;j++)for(int k=0;k<3;k++)s[i][j][k]=0;
    }
    for(int i=1,u,v;i<n;i++)scanf("%d%d",&u,&v),g[u].pb(v),g[v].pb(u);
    scanf("%s%s",c1+1,c2+1);dfs(1,0),S=0,dfs2(1);
    int ans=0,X=-(s[1][0][1]-s[1][1][2]-s[1][1][1]),Y=s[1][0][2]+s[1][1][2];
    if(X<0){puts("0");return;}
    for(int i=2;i<=n;i++)a_[i]=-(s[i][0][1]-s[i][1][2]-s[i][1][1]),b_[i]=s[i][0]
[2]+s[i][1][2];
    q1.ks(X,Y),q2.ks(X-1,Y-1);
    for(int i=2;i<=n;i++){
        int u=be[i],a=a_[u],b=b_[u];
        q1.doi(a,b),q2.doi(a-1,b-1);
        (ans-=1ll*q2.f*b%mod)%=mod;
        (ans+=1ll*q1.f*a%mod)%=mod;
    }
    ans=1ll*ans*2%mod;
    for(int i=2;i<=n;i++){
        (ans-=1ll*a_[i]*C(Y,X)%mod)%=mod;
        (ans+=1ll*b_[i]*C(Y-1,X-1)%mod)%=mod;
    }
    printf("%d\n",(ans+mod)%mod);
}
int main(){
    doi();
    int T;scanf("%d",&T);while(T--)sol();
    return 0;
}
+
```