

[比赛链接](#)

## 补题情况

题目	蒋贤蒙	王赵安	王智彪
C	0	0	0
D	0	0	0
E	2	0	0
H	2	0	0
J	2	0	0
M	0	0	0

## 题解

### E. Easy Math Problem

#### 题意

$$f(i,j) = \begin{cases} 0, & i=0 \\ a, & i=1 \\ b \times f(i-1,j) + c \times f(i-2,j), & 2 \leq i \leq j \\ d \times f(i-1,j) + e \times f(i-2,j), & i > j \end{cases}$$

#### 题解

设

$$A = \begin{pmatrix} b & 1 \\ c & 0 \end{pmatrix}, B = \begin{pmatrix} d & 1 \\ e & 0 \end{pmatrix}$$

不难发现

$$(f(i+j,j), f(i+j-1,j)) = (a,0)A^{i-1}B^j$$

设  $S(i) = \sum_{j=1}^n \binom{i+j}{j} A^{i-1} B^j$  于是有

$$\begin{aligned} S(i+1) &= \sum_{j=1}^n \binom{i+j+1}{j} A^i B^j \\ &= \sum_{j=1}^n \left( \binom{i+j}{j} + \binom{i+j}{j+1} \right) A^i B^j \\ &= AS(i) + \sum_{j=0}^{n-1} \binom{i+j}{j} A^{i-1} B^{j+1} \\ &= AS(i) + \left( S(i+1) + A^{i-1} B^{n+1} \right) B \end{aligned}$$

移项，得  $S(i+1) = \left( AS(i) + A^i B^{n+1} \right) (E-B)^{-1}$  不难发现  $|E-B| = 1-d-e \neq 0$  所以  $(E-B)$  可逆。通过预处理上式可实现  $O(n)$  递推。

```

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 frac[MAXN<<1],invf[MAXN<<1];
int C(int n,int m){
    return 1LL*frac[n]*invf[m]%mod*invf[n-m]%mod;
}
void Init(){
    int n=MAXN<<1;
    frac[0]=1;
    _for(i,1,n)
        frac[i]=1LL*frac[i-1]*i%mod;
    invf[n-1]=quick_pow(frac[n-1],mod-2);
    for(int i=n-1;i;i--)
        invf[i-1]=1LL*invf[i]*i%mod;
}
struct Matrix{
    int a[2][2];
    Matrix(int a00=0,int a01=0,int a10=0,int a11=0){
        a[0][0]=a00;
        a[0][1]=a01;
        a[1][0]=a10;
        a[1][1]=a11;
    }
    Matrix operator * (const Matrix &b)const{
        Matrix c;
        _for(i,0,2)_for(j,0,2)
            c.a[i][j]=(1LL*a[i][0]*b.a[0][j]+1LL*a[i][1]*b.a[1][j])%mod;
        return c;
    }
    Matrix operator * (const int b)const{
        Matrix c;
        _for(i,0,2)_for(j,0,2)
            c.a[i][j]=1LL*a[i][j]*b%mod;
        return c;
    }
    Matrix operator + (const Matrix &b)const{
        Matrix c;
        _for(i,0,2)_for(j,0,2)
            c.a[i][j]=(a[i][j]+b.a[i][j])%mod;
        return c;
    }
    Matrix operator - (const Matrix &b)const{
        Matrix c;
        _for(i,0,2)_for(j,0,2)
            c.a[i][j]=(a[i][j]+mod-b.a[i][j])%mod;
        return c;
    }
};
```

```

    }
}A[MAXN],B[MAXN],f[MAXN];
Matrix Inv(Matrix mat){
    static int temp[2][4];
    _for(i,0,2)_for(j,0,2)
    temp[i][j]=mat.a[i][j];
    temp[0][2]=1,temp[0][3]=0;
    temp[1][2]=0,temp[1][3]=1;
    _for(i,0,2){
        int pos=-1;
        _for(j,i,2){
            if(temp[j][i]){
                pos=j;
                break;
            }
        }
        if(pos!=i)swap(temp[i],temp[pos]);
        for(int j=3;j>=i;j--)
        temp[i][j]=1LL*temp[i][j]*quick_pow(temp[i][i],mod-2)%mod;
        _for(j,0,2){
            if(j==i)continue;
            for(int k=3;k>=i;k--)
            temp[j][k]=(temp[j][k]-1LL*temp[j][i]*temp[i][k])%mod;
        }
    }
    Matrix ans;
    _for(i,0,2)_for(j,0,2)
    ans.a[i][j]=(temp[i][j+2]+mod)%mod;
    return ans;
}
void solve(){
    int
n=read_int(),a=read_int(),b=read_int(),c=read_int(),d=read_int(),e=read_int
();
    A[0]=B[0]=Matrix(1,0,0,1);
    A[1]=Matrix(b,1,c,0);
    B[1]=Matrix(d,1,e,0);
    _rep(i,1,n){
        A[i+1]=A[i]*A[1];
        B[i+1]=B[i]*B[1];
    }
    Matrix div=Inv(B[0]-B[1]);
    f[1]=Matrix();
    _rep(i,1,n)
    f[1]=f[1]+B[i]*(i+1);
    _for(i,1,n)
    f[i+1]=(A[1]*f[i]-A[i]*B[n+1]*C(i+n+1,i+1)+A[i]*B[1])*div;
    Matrix ans=Matrix();
    _rep(i,1,n)
    ans=ans+f[i];
    enter(1LL*ans.a[0][0]*a%mod);
}

```

```
}  
int main(){  
    Init();  
    int T=read_int();  
    while(T--)  
        solve();  
    return 0;  
}
```

## J. Random Walk

### 题意

给定  $n$  个点  $m$  条边，每条边两个权值  $(a_i, b_i)$  在  $t$  时刻经过第  $i$  条边的收益为  $\max(\lfloor \frac{a_i}{2^t} \rfloor, b_i)$  且一条边经过多次收益也计算多次。

每个点一个点权  $w_i$  起点位于点  $s$  的概率为  $\frac{w_s}{\sum_{i=1}^{n-1} w_i}$  每个时间单位人物从当前点等概率移动到相邻点，且移动到点  $n$  后游戏结束。

问人物的期望收益，然后接下来两种修改操作，每次修改后再次询问人物收益：

1. 将某条边  $(u, v)$  的边权修改为  $(a, b)$  保证这条边原图中存在
2. 将某个点的  $w_i$  修改

数据保证  $1 \leq b_i \leq a_i \leq 100$  且第二类修改操作不超过  $n$  个。

### 题解

设  $\text{dp}(u, i)$  表示  $t=i$  时人物位于点  $u$  的概率  $E(u)$  表示人物经过点  $u$  的期望次数  $G(u)$  表示与  $u$  相邻的边  $p(u) = \frac{w_u}{\sum_{i=1}^{n-1} w_i}$

不难发现  $t \geq 6$  后每条边收益一定等于  $b_i$  于是答案等于

$$\sum_{u=1}^{n-1} \frac{1}{\deg(u)} \sum_{e_i \in G(u)} \sum_{t=0}^{+\infty} \max(\lfloor \frac{a_i}{2^t} \rfloor, b_i) \text{dp}(u, t) = \sum_{u=1}^{n-1} \frac{1}{\deg(u)} \sum_{e_i \in G(u)} \left( b_i \left( E(u) - \sum_{t=0}^5 \text{dp}(u, t) \right) + \sum_{t=0}^5 \max(\lfloor \frac{a_i}{2^t} \rfloor, b_i) \text{dp}(u, t) \right)$$

关于  $\text{dp}(u, t) (0 \leq t \leq 5)$  可以  $O(6m)$  暴力计算，接下来考虑计算  $E(u)$

考虑建立有向图，如果原图中边  $(u, v)$  满足  $u, v \neq n$  则连边  $u \rightarrow v (w = \frac{1}{\deg(u)})$ ,  $v \rightarrow u (w = \frac{1}{\deg(v)})$

否则，假设  $v = n$  则只连边  $u \rightarrow v (w = \frac{1}{\deg(u)})$  同时建立超级源点  $s$  连边  $s \rightarrow u (w = p(u))$

对每个点  $u$  考虑所有入边  $v_1 \rightarrow u, v_2 \rightarrow u \dots v_k \rightarrow u$  不难发现  $E(u) = \sum_{i=1}^k w_k E(v_k)$  初始条件  $E(s) = 1$

于是上述方程可以表示成如下矩阵：

$$\begin{bmatrix} c & c & c & c & | & c & 1 & a_{1,2} & \cdots & a_{1,n-1} & p(1) \\ a_{2,1} & 1 & \cdots & a_{2,n-1} & p(2) & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & 1 & p(n-1) & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$O(n^3)$  即可计算出所有  $E(u)$  因此计算初始答案复杂度为  $O(n^3+6m)$

接下来考虑修改操作，发现第一种修改操作对  $\text{dp}(u,t), E(u)$  均无影响，如果  $u \neq n$  对答案影响为

$$\frac{1^{\deg(u)} \left( b_i \left( E(u) - \sum_{t=0}^5 \text{dp}(u,t) \right) + \sum_{t=0}^5 \max(\lfloor \frac{a_i}{2^t} \rfloor, b_i) \text{dp}(u,t) \right)}{b_i}$$

同理  $v \neq n$  时也产生类似影响，因此单次处理复杂度  $O(6)$

第二种操作对  $\text{dp}(u,t), E(u)$  均产生影响，同样  $\text{dp}(u,t)$  可以暴力计算，但  $E(u)$  重新计算成本过高。

注意到  $E(u)$  的增广矩阵仅有增广部分被修改，因此可以预处理出原矩阵的逆  $A^{-1}$  则有

$$\begin{pmatrix} E(1) \\ E(2) \\ \vdots \\ E(n-1) \end{pmatrix} = A^{-1} \begin{pmatrix} p(1) \\ p(2) \\ \vdots \\ p(n-1) \end{pmatrix}$$

因此可以  $O(n^2+6m)$  重新计算一次答案。总时间复杂度  $O(n^3+6nm+6q)$

```
const int MAXN=505,MAXM=1e4+5,MAXT=5,mod=998244353;
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 w[MAXN],deg[MAXN],inv[MAXN];
int a[MAXN][MAXN],b[MAXN][MAXN],c[MAXN][MAXN],d[MAXN][MAXN],E[MAXN];
int dp[8][MAXN];
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;
}
void calc1(int n){
    static int temp[MAXN][MAXN<<1];
    n--;
    _rep(i,1,n){
        _rep(j,1,n){
```

```
        if(c[i][j])
            temp[i][j]=mod-inv[deg[j]];
        else
            temp[i][j]=0;
    }
    temp[i][i]=1;
    temp[i][i+n]=1;
}
_rep(i,1,n){
    int pos=-1;
    _rep(j,i,n){
        if(temp[j][i]){
            pos=j;
            break;
        }
    }
    if(pos!=i)swap(temp[i],temp[pos]);
    int div=quick_pow(temp[i][i],mod-2);
    for(int j=n*2;j>=i;j--)
        temp[i][j]=1LL*temp[i][j]*div%mod;
    _rep(j,1,n){
        if(j==i)continue;
        for(int k=n*2;k>=i;k--)
            temp[j][k]=(temp[j][k]-1LL*temp[j][i]*temp[i][k])%mod;
    }
}
_rep(i,1,n)_rep(j,1,n)
d[i][j]=(temp[i][j+n]+mod)%mod;
}
void add_edge(int u,int v,int &ans){
    _rep(t,0,MAXT)
    ans=(ans+1LL*dp[t][u]*inv[deg[u]]%mod*max(a[u][v]>>t,b[u][v]))%mod;
    ans=(ans+1LL*E[u]*inv[deg[u]]%mod*b[u][v])%mod;
}
void del_edge(int u,int v,int &ans){
    _rep(t,0,MAXT)
    ans=(ans-1LL*dp[t][u]*inv[deg[u]]%mod*max(a[u][v]>>t,b[u][v]))%mod;
    ans=(ans-1LL*E[u]*inv[deg[u]]%mod*b[u][v])%mod;
    ans=(ans+mod)%mod;
}
int calc2(int n){
    int ans=0,s=0;
    mem(dp,0);
    n--;
    _rep(i,1,n)s+=w[i];
    s=quick_pow(s,mod-2);
    _rep(i,1,n){
        dp[0][i]=1LL*w[i]*s%mod;
        E[i]=0;
        _rep(j,1,n)
```

```

    E[i]=(E[i]+1LL*w[j]*d[i][j])%mod;
    E[i]=1LL*E[i]*s%mod;
    E[i]=(E[i]+mod-dp[0][i])%mod;
}
_rep(t,1,MAXT){
    _rep(u,1,n){
        for(int i=head[u];i;i=edge[i].next){
            int v=edge[i].to;
            dp[t][v]=(dp[t][v]+1LL*dp[t-1][u]*inv[deg[u]])%mod;
        }
    }
    _rep(u,1,n)
    E[u]=(E[u]+mod-dp[t][u])%mod;
}
_rep(u,1,n){
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        add_edge(u,v,ans);
    }
}
return ans;
}
int main(){
    int n=read_int(),m=read_int(),q=read_int();
    _for(i,1,n)
    w[i]=read_int();
    while(m--){
        int u=read_int(),v=read_int();
        a[u][v]=a[v][u]=read_int();
        b[u][v]=b[v][u]=read_int();
        c[u][v]=c[v][u]=1;
        Insert(u,v);Insert(v,u);
        deg[u]++;deg[v]++;
    }
    inv[1]=1;
    _rep(i,2,n)
    inv[i]=1LL*(mod-mod/i)*inv[mod%i]%mod;
    calc1(n);
    int ans=calc2(n);
    enter(ans);
    while(q--){
        int opt=read_int();
        if(opt==1){
            int u=read_int(),v=read_int();
            if(u>v)swap(u,v);
            del_edge(u,v,ans);
            if(v!=n)del_edge(v,u,ans);
            a[u][v]=a[v][u]=read_int();
            b[u][v]=b[v][u]=read_int();
            add_edge(u,v,ans);
            if(v!=n)add_edge(v,u,ans);
        }
    }
}

```

```
    }  
    else{  
        int u=read_int();  
        w[u]=read_int();  
        ans=calc2(n);  
    }  
    enter(ans);  
}  
return 0;  
}
```

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:%E7%BB%84%E9%98%9F%E8%AE%AD%E7%BB%83%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95:contest17&rev=1630419614](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%BB%84%E9%98%9F%E8%AE%AD%E7%BB%83%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95:contest17&rev=1630419614)

Last update: 2021/08/31 22:20