NOIP2023 模拟赛题解

> 南外张超教练 2023/10/19 13:58:31
> 我一直觉得 `#define int long long` 是很 low 的行为，不知道大家怎么看

B. 排序

考虑维护每个 a_i 在第 j 个时刻所在的位置 p_j。

当我们想要加入某个 $a_i = 1$ 时，找到第一个 pos 满足 $p_{pos} = i - j - 1$，若没有设为 m:

- 对于 $j \le pos$，设 $p_j \leftarrow i - j$。
- 对于 $j > pos$，则 $p_j \leftarrow p_{j-1} + 1$。

注意转移的顺序。

考虑维护 $p_j + j$ 的差分数组 d_j:

找到第一个 pos 满足 $\sum_{k=0}^{pos-1} d_k = i - 1$，若不存在则设为 $m + 1$。

此时 $d_0 = i, d_{pos} = 1$。

- 对于 $1 \le j < pos$，$d_j \leftarrow 0$。
- 对于 $pos < j \le m$，$d_j \leftarrow d_{j-1}$。

然后我们发现每次至多加入 $\mathcal{O}(1)$ 个新的非 0 的 d_j，因此考虑用一个双端队列维护所有 $d_j \neq 0$ 的 (j, d_j)。

找 pos 的时候可以暴力弹出队头，均摊复杂度不变，$d_j \leftarrow d_{j-1}$ 相当于支持给所有下标 $+1$，维护一个 j 偏移量的标记即可。

时间复杂度 $\mathcal{O}(n)$。

参考代码

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN=3e6+5;
int a[MAXN],b[MAXN];
signed main() {
    freopen("sort.in","r",stdin);
    freopen("sort.out","w",stdout);
    int n,m;
    scanf("%d%d",&n,&m);
    deque <array<int,2>> Q;
    for(int i=1,flg=0,s=0,dx=0;i<=n;++i) {
        scanf("%d",&a[i]);
        if(!a[i]) continue;
        if(!flg) {
            Q.push_back({0,i}),s=m+1;flg=1;
            for(int j=i;j<=m;++j) Q.push_back({j,1});
        } else {
            int pos=m;
            for(int t=0;!Q.empty();) {
                int tmp=Q.front()[0]+dx;
                t+=Q.front()[1],s-=Q.front()[1],Q.pop_front();
                if(t==i-1) { pos=tmp; break; }
            }
            ++pos;
            if(pos<=m) {
                if(Q.back()[0]+dx==m) s-=Q.back()[1],Q.pop_back();
                ++dx;
                Q.push_front({pos-dx,1}),++s;
            }
            Q.push_front({-dx,i}),s+=i;
        }
        b[s-m]=1;
    }
    for(int i=1;i<=n;++i) printf("%d ",b[i]);
    puts("");
}
```

C. 洪水

显然首先转成计数方案，然后用反面考虑的思想转化成计数多少种取边权的方案使得每个点都至少被一个洪水覆盖。

考虑一个朴素的 dp，状态 $dp_{u,\mathbf{S},\mathbf{T}}$ 表示以 u 为根的子树全被覆盖的情况下：可以从子树中移动到 u 的洪水构成 $\mathbf{S}$，需要从外界进入 u 以帮助淹没 u 子树的洪水~~构成 $\mathbf{T}$~~。

考虑一个优化，注意到 $\mathbf{S},\mathbf{T}$ 中我们实际只关心其中最大的 h_i，因此可以离散化 h_i，记对应的 h_i 去重并排序后得到 $v_1,v_2,\ldots,v_k$，则可以用 $dp_{u,s,t}$ 表示 $\mathbf{S}$ 中 h_i 最大值为 v_s，$\mathbf{T}$ 中 h_i 最大值为 v_t 的方案数。

注意到一个观察：若 $\mathbf{S}$ 非空，则 $\mathbf{T}$ 一定为空，否则要么 $\mathbf{S}$ 中的洪水可以去覆盖对应节点，要么 $v_s<v_t$ 导致 $\mathbf{S}$ 中的人去外界覆盖不如直接用 $\mathbf{T}$ 中的洪水，同理可证 $\mathbf{T}$ 非空时 $\mathbf{S}$ 一定为空。

因此我们可以把原来的状态拆成两个更简单的状态 $f_{u,s},g_{u,t}$ 含义同上，分别代表 $\mathbf{S}$ 非空或 $\mathbf{T}$ 非空的情况。

考虑合并两个状态 $(u,x),(v,y)$，枚举边权 w_e 所在区间 $[v_{w-1},v_w)$，求出 $len_w=|[l_i,r_i]\cap(w_{w-1},v_w]|$，表示每种情况对应的实际边权取值数量，然后分类讨论，分别考虑如下转移：

- 转移 $f_{u,x}\oplus f_{v,y}$：
  - 若 $w\le y$，说明 y 可以移动到 u 上，则 $f_{u,\max(x,y)}\leftarrow f_{u,x}\times f_{v,y}\times len_w$。
  - 否则，说明能够移动到 u 上的只有 x，则 $f_{u,x}\leftarrow f_{u,x}\times f_{V,y}\times len_w$。
- 转移 $f_{u,x}\oplus g_{v,y}$：
  - 若 $x\ge w$ 且 $x\ge y$，说明 x 移动到 v 上后可以解决问题，则 $f_{u,x}\leftarrow f_{u,x}\times g_{v,y}\times len_w$。
  - 否则，说明需要一个外界的点来满足 v 子树的需求，则 $g_{u,\max(y,w)}\leftarrow f_{u,x}\times g_{v,y}\times len_w$。
- 转移 $g_{u,x}\oplus f_{v,y}$：
  - 若 $y\ge w$ 且 $y\ge x$，说明 y 移动到 u 上后可解决问题，则 $f_{u,y}\leftarrow g_{u,x}\times f_{v,y}\times len_w$。
  - 否则说明需要一个外界的点来满足 u 子树的需求，则 $g_{u,x}\leftarrow g_{u,x}\times f_{v,y}\times len_w$。
- 转移 $g_{u,x}\oplus g_{v,y}$：
  - 说明至少需要一个外界的点满足 u 子树的需求，并经过 e 去满足 v 的需求，则 $g_{u,\max(x,y,w)}\leftarrow g_{u,x}\times g_{v,y}\times len_w$。

因此朴素地枚举 x,y,w 可以做到 $\mathcal{O}(nm^3)$，考虑优化。

观察七种状态转移方程，考虑枚举被更新的下标 i，再观察 (x,y,w) 与 i 的关系，容易发现该问题可以变成若干组前后缀和来处理，因此分别考虑每种情况用前缀和优化即可。

建议手写 `modint` 类以减小码量和程序常数（手动实现加减法的取模优化）。

时间复杂度 $\mathcal{O}(nm)$。

参考代码

$\mathcal{O}(nm^3)$ 实现，60pts。

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int MAXN=2005,MOD=998244353,INF=2e9;
inline int ksm(int a,int b=MOD-2,int p=MOD) {
    int ret=1;
    while(b) ret=(b&1?ret*a%p:ret),a=a*a%p,b=b>>1;
    return ret;
}
struct mint {
    int v;
    mint(int w=0) { v=w<MOD?w:w%MOD; }
    inline friend mint operator +(mint a,mint b) { return a.v+b.v>=MOD?a.v+b.v-MOD:a.v+b.v; }
    inline friend mint operator +( int a,mint b) { return mint(a)+b; }
    inline friend mint operator +(mint a, int b) { return a+mint(b); }
    inline void operator +=(mint o) { v=v+o.v>=MOD?v+o.v-MOD:v+o.v; }
    inline void operator +=(int o) { v=v+o>=MOD?v+o-MOD:v+o; }

    inline friend mint operator -(mint a,mint b) { return a.v>=b.v?a.v-b.v:a.v+MOD-b.v; }
    inline friend mint operator -( int a,mint b) { return mint(a)-b; }
    inline friend mint operator -(mint a, int b) { return a-mint(b); }
    inline void operator -=(mint o) { v=v>=o.v?v-o.v:v+MOD-o.v; }
    inline void operator -=(int o) { v=v>=o?v-o:v+MOD-o; }

    inline friend mint operator *(mint a,mint b) { return a.v*b.v%MOD; }
    inline friend mint operator *( int a,mint b) { return mint(a)*b; }
    inline friend mint operator *(mint a, int b) { return a*mint(b); }
    inline void operator *=(mint o) { v=v*o.v%MOD; }
    inline void operator *=(int o) { v=v*o%MOD; }
};
struct Edge { int v,l,r; };
vector <Edge> G[MAXN];
mint f[MAXN][MAXN],g[MAXN][MAXN],tf[MAXN],tg[MAXN];
//f[u][x]: cover tree(u), free x
//g[u][x]: cover tree(u), need x
int n,m,z[MAXN],vals[MAXN];
inline void dfs(int u,int fa) {
    for(int i=0;i<=m;++i) f[u][i]=g[u][i]=0;
    if(z[u]) f[u][z[u]]=1;
    else g[u][1]=1;
    for(auto e:G[u]) if(e.v!=fa) {
        int v=e.v;
        dfs(v,u);
        for(int i=0;i<=m;++i) tf[i]=tg[i]=0;
        for(int x=0;x<=m;++x) for(int y=0;y<=m;++y) {
```

```
            for(int w=1;w<=m;++w) { //v in [vals w-1,vals w)
                auto inter=[&](int ul,int ur,int vl,int vr) {
                    if(ul>vl) swap(ul,vl),swap(ur,vr);
                    return ur<vl?0:min(ur,vr)-vl+1;
                };
                int c=inter(e.l,e.r,vals[w-1],vals[w]-1);
                if(!c) continue;
                if(w<=y) tf[max(x,y)]+=f[u][x]*f[v][y]*c;
                else tf[x]+=f[u][x]*f[v][y]*c;

                if(w<=x&&x>=y) tf[x]+=f[u][x]*g[v][y]*c;
                else tg[max(y,w)]+=f[u][x]*g[v][y]*c;

                if(w<=y&&y>=x) tf[y]+=g[u][x]*f[v][y]*c;
                else tg[x]+=g[u][x]*f[v][y]*c;

                tg[max({x,y,w})]+=g[u][x]*g[v][y]*c;
            }
        }
        swap(f[u],tf),swap(g[u],tg);
    }
}
signed main() {
    freopen("flood.in","r",stdin);
    freopen("flood.out","w",stdout);
    scanf("%lld%lld",&n,&m);
    for(int i=1;i<=n;++i) z[i]=0,G[i].clear();
    mint p=1;
    for(int i=1,u,v,l,r;i<n;++i) {
        scanf("%lld%lld%lld%lld",&u,&v,&l,&r),p*=r-l+1;
        G[u].push_back({v,l,r}),G[v].push_back({u,l,r});
    }
    for(int i=1,x,h;i<=m;++i) scanf("%lld%lld",&x,&h),z[x]=max(z[x],h);
    for(int i=1;i<=n;++i) vals[i]=z[i];
    sort(vals,vals+n+1),m=unique(vals,vals+n+1)-vals;
    vals[m]=INF;
    for(int i=1;i<=n;++i) z[i]=lower_bound(vals,vals+m,z[i])-vals;
    dfs(1,0);
    mint ans=0;
    for(int i=1;i<=m;++i) ans+=f[1][i];
    ans=1-ans*ksm(p.v);
    printf("%lld\n",ans.v);
    return 0;
}
```

$\mathcal{O}(nm)$ 实现，100pts。

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int MAXN=2005,MOD=998244353,INF=2e9;
inline int ksm(int a,int b=MOD-2,int p=MOD) {
    int ret=1;
    while(b) ret=(b&1?ret*a%p:ret),a=a*a%p,b=b>>1;
    return ret;
}
struct mint {
    int v;
    mint(int w=0) { v=w<MOD?w:w%MOD; }
    inline friend mint operator +(mint a,mint b) { return a.v+b.v>=MOD?a.v+b.v-MOD:a.v+b.v; }
    inline friend mint operator +( int a,mint b) { return mint(a)+b; }
    inline friend mint operator +(mint a, int b) { return a+mint(b); }
    inline void operator +=(mint o) { v=v+o.v>=MOD?v+o.v-MOD:v+o.v; }
    inline void operator +=(int o) { v=v+o>=MOD?v+o-MOD:v+o; }

    inline friend mint operator -(mint a,mint b) { return a.v>=b.v?a.v-b.v:a.v+MOD-b.v; }
    inline friend mint operator -( int a,mint b) { return mint(a)-b; }
    inline friend mint operator -(mint a, int b) { return a-mint(b); }
    inline void operator -=(mint o) { v=v>=o.v?v-o.v:v+MOD-o.v; }
    inline void operator -=(int o) { v=v>=o?v-o:v+MOD-o; }

    inline friend mint operator *(mint a,mint b) { return a.v*b.v%MOD; }
    inline friend mint operator *( int a,mint b) { return mint(a)*b; }
    inline friend mint operator *(mint a, int b) { return a*mint(b); }
    inline void operator *=(mint o) { v=v*o.v%MOD; }
    inline void operator *=(int o) { v=v*o%MOD; }
};
struct Edge { int v,l,r; };
vector <Edge> G[MAXN];
mint f[MAXN][MAXN],g[MAXN][MAXN],tf[MAXN],tg[MAXN];
//f[u][x]: cover tree(u), free x
//g[u][x]: cover tree(u), need x
mint su[MAXN],sv[MAXN],sl[MAXN]; //prefix sums
int n,m,z[MAXN],vals[MAXN],len[MAXN];
inline void dfs(int u,int fa) {
    for(int i=0;i<=m;++i) f[u][i]=g[u][i]=0;
    if(z[u]) f[u][z[u]]=1;
    else g[u][1]=1;
    for(auto e:G[u]) if(e.v!=fa) {
        int v=e.v;
        mint S;
        dfs(v,u);
        for(int i=0;i<=m;++i) tf[i]=tg[i]=0;
```

```
        for(int w=0;w<=m;++w) {
            auto inter=[&](int ul,int ur,int vl,int vr) {
                if(ul>vl) swap(ul,vl),swap(ur,vr);
                return ur<vl?0:min(ur,vr)-vl+1;
            };
            len[w]=w?inter(e.l,e.r,vals[w-1],vals[w]-1):0;
            sl[w]=w?inter(e.l,e.r,0,vals[w]-1):0;
        }

        //w<=y: tf[max(x,y)]+=f[u][x]*f[v][y]*len[w]
        //w> y: tf[x]+=f[u][x]*f[v][y]*len[w]
        S=0;
        for(int i=0;i<=m;++i) {
            su[i]=i?su[i-1]:0,sv[i]=i?sv[i-1]:0;
            sv[i]+=sl[i]*f[v][i],su[i]+=f[u][i];
            S+=(sl[m]-sl[i])*f[v][i];
        }
        for(int i=0;i<=m;++i) {
            su[i]*=sv[i];
            tf[i]+=(i?su[i]-su[i-1]:su[i])+S*f[u][i];
        }

        //max(w,y)<=x: tf[x]+=f[u][x]*g[v][y]*len[w]
        //max(w,y)> x: tg[max(y,w)]+=f[u][x]*g[v][y]*len[w]
        for(int i=0;i<=m;++i) {
            su[i]=i?su[i-1]:0,sv[i]=i?sv[i-1]:0;
            su[i]+=f[u][i],sv[i]+=g[v][i];
        }
        for(int i=0;i<=m;++i) {
            sv[i]*=sl[i];
            tf[i]+=f[u][i]*sv[i];
            if(i) tg[i]+=(sv[i]-sv[i-1])*su[i-1];
        }

        //max(x,w)<=y: tf[y]+=g[u][x]*f[v][y]*len[w]
        //max(x,w)> y: tg[x]+=g[u][x]*f[v][y]*len[w]
        S=0; //w>y
        mint t=0;
        for(int i=0;i<=m;++i) {
            su[i]=i?su[i-1]:0,sv[i]=i?sv[i-1]:0;
            su[i]+=g[u][i],sv[i]+=f[v][i]*sl[i]; //w<=y<=i
            S+=len[i]*t,t+=f[v][i];
        }
        for(int i=0;i<=m;++i) {
            tf[i]+=f[v][i]*su[i]*sl[i];
            tg[i]+=g[u][i]*(i?S+sv[i-1]:S);
        }

        //tg[max(x,y,w)]+=g[u][x]*g[v][y]*len[w]
        for(int i=0;i<=m;++i) {
            su[i]=i?su[i-1]:0,sv[i]=i?sv[i-1]:0;
```

```
            su[i]+=g[u][i],sv[i]+=g[v][i];
        }
        for(int i=0;i<=m;++i) {
            su[i]*=sv[i]*sl[i];
            tg[i]+=(i?su[i]-su[i-1]:su[i]);
        }
        swap(f[u],tf),swap(g[u],tg);
    }
}
signed main() {
    freopen("flood.in","r",stdin);
    freopen("flood.out","w",stdout);
    scanf("%lld%lld",&n,&m);
    mint p=1;
    for(int i=1,u,v,l,r;i<n;++i) {
        scanf("%lld%lld%lld%lld",&u,&v,&l,&r),p*=r-l+1;
        G[u].push_back({v,l,r}),G[v].push_back({u,l,r});
    }
    for(int i=1,x,h;i<=m;++i) scanf("%lld%lld",&x,&h),z[x]=max(z[x],h);
    for(int i=1;i<=n;++i) vals[i]=z[i];
    sort(vals,vals+n+1),m=unique(vals,vals+n+1)-vals;
    vals[m]=INF;
    for(int i=1;i<=n;++i) z[i]=lower_bound(vals,vals+m,z[i])-vals;
    dfs(1,0);
    mint ans=0;
    for(int i=1;i<=m;++i) ans+=f[1][i];
    ans=1-ans*ksm(p.v);
    printf("%lld\n",ans.v);
    return 0;
}
```

D. 流量

先转最小割，考虑树上的情况，答案就是路径最小值，然后考虑环，就是把两段路径最小值加起来即可。

考虑建圆方树刻画仙人掌，圆点 u 到父亲方点 c 的边权值定义为环上 $u \to fa(c)$ 的两段路径最小值的和，这一部分对每个环建线段树维护。

方点到父亲的权值定义为 ∞，那么答案就是圆方树上把 $u \to v$ 路径所有边权取 $\min$，树剖维护，特殊处理 LCA 是方点的情况。

但此时在修改一条边时，要把这个环上所有不是根的点的边权都修改，复杂度难以接受。

考虑邻域修改的经典技巧，只修改重边权值，若 $u \to fa(c)$ 是轻边则权值设为 ∞，在求 LCA 时统计这条边的贡献。

时间复杂度 $\mathcal{O}((n+q)\log^2 n)$。

参考代码

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN=2e5+5,inf=2e9;
struct SegmentTree {
    int N;
    vector <int> tr;
    inline void init(int n) { N=1<<__lg(2*n+5),tr.resize(2*N,inf); }
    inline void Modify(int u,int v) {
        for(tr[u+=N]=v,u>>=1;u;u>>=1) tr[u]=min(tr[u<<1],tr[u<<1|1]);
    }
    inline int Query(int l,int r) {
        int ans=inf;
        for(l+=N-1,r+=N+1;l^r^1;l>>=1,r>>=1) ans=l&1?ans:min(ans,tr[l^1]),ans=r&1?
min(ans,tr[r^1]):ans;
        return ans;
    }
}   seg[MAXN];
vector <int> T[MAXN],G[MAXN];
int vers[MAXN][2];
unordered_map <int,int> idx[MAXN],cap[MAXN];
int n,m,q,tot;
int vrk[MAXN],erk[MAXN],vcy[MAXN],ecy[MAXN],len[MAXN];
int tim[MAXN],low[MAXN],tcnt,stk[MAXN],tp;
inline void tarjan(int u) {
    tim[u]=low[u]=++tcnt,stk[++tp]=u;
    for(int v:T[u]) {
        if(!tim[v]) {
            tarjan(v),low[u]=min(low[u],low[v]);
            if(low[v]==tim[u]) {
                auto link=[&](int s,int t) { G[s].push_back(t),G[t].push_back(s); };
                if(stk[tp]==v) link(u,v),--tp;
                else {
                    int k; ++tot;
                    do link(tot,k=stk[tp--]); while(k!=v);
                    link(u,tot),len[tot]=G[tot].size(),seg[tot].init(len[tot]);
                    for(int i=0;i<len[tot];++i) {
                        int x=G[tot][i],y=G[tot][(i+1)%len[tot]];
                        vrk[x]=erk[idx[x][y]]=i+1,vcy[x]=ecy[idx[x][y]]=tot;
                        seg[tot].Modify(i+1,cap[x][y]);
                    }
                }
            }
        } else low[u]=min(low[u],tim[v]);
    }
}
int siz[MAXN],Fa[MAXN],hson[MAXN],dep[MAXN],top[MAXN],dfn[MAXN],dcnt;
int up[MAXN][20];
inline void dfs0(int u,int fa) {
```

```
    siz[u]=1,up[u][0]=Fa[u]=fa,dep[u]=dep[fa]+1;
    for(int k=1;k<20;++k) up[u][k]=up[up[u][k-1]][k-1];
    for(int v:G[u]) if(v^fa) {
        dfs0(v,u),siz[u]+=siz[v];
        if(siz[v]>siz[hson[u]]) hson[u]=v;
    }
}
inline int dist(int x,int y) {
    if(vrk[x]>vrk[y]) swap(x,y);
    int t=vcy[x],out=min(seg[t].Query(vrk[y],len[t]),seg[t].Query(1,vrk[x]-1));
    return seg[t].Query(vrk[x],vrk[y]-1)+out;
}
inline void dfs1(int u,int rt) {
    top[u]=rt,dfn[u]=++dcnt;
    if(u<=n&&Fa[u]) {
        if(Fa[u]<=n) seg[0].Modify(dfn[u],cap[u][Fa[u]]);
        else if(u==hson[Fa[u]]) seg[0].Modify(dfn[u],dist(u,Fa[Fa[u]]));
    }
    if(hson[u]) dfs1(hson[u],rt);
    for(int v:G[u]) if(v!=Fa[u]&&v!=hson[u]) dfs1(v,v);
}
inline void modify(int e,int w) {
    if(!ecy[e]) {
        int u=vers[e][0],v=vers[e][1];
        if(Fa[v]==u) swap(u,v);
        seg[0].Modify(dfn[u],w);
    } else {
        int u=ecy[e],x=hson[u];
        seg[u].Modify(erk[e],w),seg[0].Modify(dfn[x],dist(x,Fa[u]));
    }
}
inline int subtree(int p,int rt) {
    for(int k=19;~k;--k) if(dep[up[p][k]]>dep[rt]) p=up[p][k];
    return p;
}
inline int query(int u,int v) {
    int ans=inf,s=u,t=v;
    while(top[u]^top[v]) {
        if(dep[top[u]]<dep[top[v]]) swap(u,v);
        ans=min(ans,seg[0].Query(dfn[top[u]],dfn[u]));
        u=top[u];
        if(u<=n&&Fa[u]>n) ans=min(ans,dist(u,Fa[Fa[u]]));
        u=Fa[u];
    }
    if(dep[u]>dep[v]) swap(u,v);
    ans=min(ans,seg[0].Query(dfn[u]+1,dfn[v]));
    if(u>n) {
        int x=subtree(s,u),y=subtree(t,u);
        return min({query(s,x),query(t,y),dist(x,y)});
    }
    return ans;
```

```
}
signed main() {
    freopen("flow.in","r",stdin);
    freopen("flow.out","w",stdout);
    int Tid;
    scanf("%d%d%d%d",&Tid,&n,&m,&q),tot=n;
    for(int i=1,u,v,w;i<=m;++i) {
        scanf("%d%d%d",&u,&v,&w);
        T[u].push_back(v),T[v].push_back(u);
        vers[i][0]=u,vers[i][1]=v,idx[u][v]=idx[v][u]=i,cap[u][v]=cap[v][u]=w;
    }
    tarjan(1),seg[0].init(tot),dfs0(1,0),dfs1(1,1);
    for(int op,x,y;q--;) {
        scanf("%d%d%d",&op,&x,&y);
        if(op==1) modify(x,y);
        else printf("%d\n",query(x,y));
    }
    return 0;
}
```