其他东西

目录

Set 暴力更改区间

```
void update(int l,int r,int x){
    auto final=make_pair(make_pair(l,r),x);
    while (l<=r){
        auto
it=POS.upper_bound(make_pair(make_pair(l,INF),0)); it--;
        auto now=*it; POS.erase(it);
        int nxtl=now.first.second+1;
        if (now.first.first<l){
            pair<int,int> remain;
            remain.first=now.first.first;
            remain.second=l-1;
            if (remain.first<=remain.second)

POS.insert(make_pair(remain,now.second));
        }
        if (now.first.second>r){
            pair<int,int> remain;
            remain.first=r+1;
            remain.second=now.first.second;
            if (remain.first<=remain.second)

POS.insert(make_pair(remain,now.second));
            nxtl=r+1;
        }
        update(now.second,-(nxtl-l));
        update(x,nxtl-l);
        l=nxtl;
    } POS.insert(final);
}
```

缩点

```
vector<int> edge[maxn],vt[maxn],kp;
bool key[maxn];
int build(int x,int fa){
    vector<int> ch;
    for (auto v:edge[x]){
        if (v==fa) continue;
        int w=build(v,x);
        if (w) ch.push_back(w);
    } if (ch.size()>=2) key[x]=1;
    if (key[x]){//virtual tree
        kp.push_back(x);
        for (auto v:ch) {
            vt[x].push_back(v);
            vt[v].push_back(x);
        } return x;
    } if (ch.size()) return ch[0];
    return 0;//no key
}
int ans; bool vis[maxn];
void dfs(int u,int s,int dep){
    int cnt=0;
    for (auto v:vt[u]) {
        if (v==s) ++cnt;
        if (v<s&&!vis[v]){
            vis[v]=1; dfs(v,s,dep+1); vis[v]=0;
        }
    } if (cnt>1||(cnt==1&&dep>2)) ans++;
}
int fa[maxn];
inline int getfa(int x){
    if (fa[x]==x) return x;
    return fa[x]=getfa(fa[x]);
}
int main(){
    int i;
    scanf("%d%d",&n,&m);
    FOR(i,1,n) fa[i]=i;
    FOR(i,1,m) {
```

```
            int u,v;
            scanf("%d%d",&u,&v);
            int x=getfa(u),y=getfa(v);
            if (x==y) {
                key[u]=key[v]=1;
                vt[u].push_back(v);
                vt[v].push_back(u);
            }else {
                fa[x]=y;
                edge[u].push_back(v);
                edge[v].push_back(u);
            }
        } key[1]=1;
        build(1,0);
        for(auto s:kp) {
            // printf("%d ",s);
            vis[s]=1; dfs(s,s,1); vis[s]=0;
        } printf("%d\n",ans/2);
}
```

费用流(快的那个)

```
// 这个好像就是zkw费用流
// 拆点后可以S向入连边，出向T连边，然后入和出就可以保持动态平衡!
// 连边是为了将"获取的"和"使用的"联系起来! 大概意思就是，使用的流量确定...
// 注意观察特殊性质
// 费用流有个"短路"的性质，如果流到这里可能会使得其他的流量减少，这个好像有点用
namespace mincostflow {
    typedef int type;
    const type INF=0x3f3f3f3f;
    struct node {
        int to; type cap,cost; int rev;
        node(int t=0,type c=0,type _c=0,int n=0):
            to(t),cap(c),cost(_c),rev(n) {};
    }; vector<node> edge[maxn];
    void addedge(int from,int to,type cap,type cost,type rcap=0) {
        edge[from].push_back(node(to,cap,cost,edge[to].size()));
        edge[to].push_back(node(from,rcap,-cost,edge[from].size()-1));
    }
    type dis[maxn];
    bool mark[maxn];
    void spfa(int s,int t,int n) {
        memset(dis+1,0x3f,n*sizeof(type));
        memset(mark+1,0,n*sizeof(bool));
        static int Q[maxn],ST,ED;
        dis[s]=0; ST=ED=0; Q[ED++]=s;
        while (ST!=ED) {
            int v=Q[ST]; mark[v]=0;
            if ((++ST)==maxn) ST=0;
            for (node &e:edge[v]) {
                if (e.cap>0&&dis[e.to]>dis[v]+e.cost) {
                    dis[e.to]=dis[v]+e.cost;
                    if (!mark[e.to]) {
                        if (ST==ED||dis[Q[ST]]<=dis[e.to]) {
                            Q[ED]=e.to,mark[e.to]=1;
                            if ((++ED)==maxn) ED=0;
                        } else {
                            if ((--ST)<0) ST+=maxn;
                            Q[ST]=e.to,mark[e.to]=1;
                        }
                    }
                }
            }
        }
    } int cur[maxn];
    type dfs(int x,int t,type flow) {
        if (x==t||!flow) return flow;
        type ret=0; mark[x]=1;
        int i;
        rep(i,cur[x],(int)edge[x].size()) {
            node &e=edge[x][i];
            if (!mark[e.to]&&e.cap) {
                if (dis[x]+e.cost==dis[e.to]) {
                    int f=dfs(e.to,t,min(flow,e.cap));
                    e.cap-=f; edge[e.to][e.rev].cap+=f;
                    ret+=f; flow-=f; cur[x]=i;
                    if (flow==0) break;
                }
            }
        } mark[x]=0;
        return ret;
    }
    pair<type,type> mincostflow(int s,int t,int n,type flow=INF) {
        type ret=0,ans=0;
        while (flow) {
            spfa(s,t,n); if (dis[t]==INF) break;
```

```
            // 这样加当前弧优化会快，我也不知道为啥
            memset(cur+1,0,n*sizeof(int));
            type len=dis[t],f;
            while ((f=dfs(s,t,flow))>0)//while也行
                ret+=f,ans+=len*f,flow-=f;
        } return make_pair(ret,ans);
    }
    void init(int n) {
        int i; FOR(i,1,n) edge[i].clear();
    }
}
```

快速乘

```
return (x*y-(long long)(x/(long double)MOD*y+1e-3)*MOD+MOD)%MOD;
```

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll mod=1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return
res;}
ll gcd(ll a,ll b) { return b?gcd(b,a%b):a;}
// head

typedef unsigned long long u64;
typedef __int128_t i128;
typedef __uint128_t u128;
int _,k;
u64 A0,A1,M0,M1,C,M;

struct Mod64 {
    Mod64():n_(0) {}
    Mod64(u64 n):n_(init(n)) {}
    static u64 init(u64 w) { return reduce(u128(w) * r2); }
    static void set_mod(u64 m) {
        mod=m; assert(mod&1);
        inv=m; rep(i,0,5) inv*=2-inv*m;
        r2=-u128(m)%m;
    }
    static u64 reduce(u128 x) {
        u64 y=u64(x>>64)-
u64((u128(u64(x)*inv)*mod)>>64);
        return ll(y)<0?y+mod:y;
    }
    Mod64& operator += (Mod64 rhs) { n_+=rhs.n_-mod; if
(ll(n_)<0) n_+=mod; return *this; }
    Mod64 operator + (Mod64 rhs) const { return
Mod64(*this)+=rhs; }
    Mod64& operator -= (Mod64 rhs) { n_-=rhs.n_; if
(ll(n_)<0) n_+=mod; return *this; }
    Mod64 operator - (Mod64 rhs) const { return
Mod64(*this)-=rhs; }
    Mod64& operator *= (Mod64 rhs)
{ n_=reduce(u128(n_)*rhs.n_); return *this; }
    Mod64 operator * (Mod64 rhs) const { return
Mod64(*this)*=rhs; }
    u64 get() const { return reduce(n_); }
    static u64 mod,inv,r2;
    u64 n_;
};
u64 Mod64::mod,Mod64::inv,Mod64::r2;

u64 pmod(u64 a,u64 b,u64 p) {
    u64 d=(u64)floor(a*(long double)b/p+0.5);
    ll ret=a*b-d*p;
    if (ret<0) ret+=p;
    return ret;
}


void bruteforce() {
    u64 ans=1;
    for (int i=0;i<=k;i++) {
        ans=pmod(ans,A0,M);
        u64 A2=pmod(M0,A1,M)+pmod(M1,A0,M)+C;
        while (A2>=M) A2-=M;
```

```
            A0=A1; A1=A2;
        }
        printf("%llu\n",ans);
}

int main() {
        for (scanf("%d",&_);_;_--) {

scanf("%llu%llu%llu%llu%llu%llu%d",&A0,&A1,&M0,&M1,&
C,&M,&k);
            Mod64::set_mod(M);
            Mod64
a0(A0),a1(A1),m0(M0),m1(M1),c(C),ans(1),a2(0);
            for (int i=0;i<=k;i++) {
                ans=ans*a0;
                a2=m0*a1+m1*a0+c;
                a0=a1; a1=a2;
            }
            printf("%llu\n",ans.get());
        }
}
```