C++ Header Files

Contents

Header.hpp

```cpp
#pragma GCC optimize("Ofast,no-stack-protector,unroll-loops")
#define ALL(v) v.begin(),v.end()
#define For(i,_) for(int i=0,i##end=_;i<i##end;++i) // [0,_)
#define FOR(i,_,__) for(int i=_,i##end=__;i<i##end;++i) // [_,__)
#define Rep(i,_) for(int i=(_)-1;i>=0;--i) // [0,_)
#define REP(i,_,__) for(int i=(__)-1,i##end=_;i>=i##end;--i) // [_,__)
typedef long long ll;
typedef unsigned long long ull;
#define V vector
#define pb push_back
#define pf push_front
#define qb pop_back
#define qf pop_front
#define eb emplace_back
```

```cpp
typedef pair<int,int> pii;
typedef pair<ll,int> pli;
#define fi first
#define se second
const int dir[4][2]={{-1,0},{0,1},{1,0},{0,-1}},inf=0x3f3f3f3f,mod=1e9+7;
const ll infl=0x3f3f3f3f3f3f3f3fll;
template<class T>inline bool ckmin(T &x,const T &y){return x>y?x=y,1:0;}
template<class T>inline bool ckmax(T &x,const T &y){return x<y?x=y,1:0;}
int init=[](){return cin.tie(nullptr)->sync_with_stdio(false),0;}();
```

ST 表.hpp

```cpp
template<class T,T(*merge)(T,T)>
struct ST{
    V<V<T>>st;
    inline ST(){}
    inline ST(const V<T> &a){
        int n=a.size(),B=__lg(n);
        V<V<T>>(B+1).swap(st);
        st[0]=a;
        FOR(i,1,B+1){
            st[i].resize(n-(1<<i)+1);
            For(j,n-(1<<i)+1)st[i][j]=merge(st[i-1][j],st[i-1][j+(1<<i-1)]);
        }
    }
    inline ST(const V<T> &a,const V<int> &pos){
        assert(a.size()==pos.size());
        int n=a.size(),B=__lg(n);
        V<V<T>>(B+1).swap(st);
        For(i,B+1){
            st[i].resize(n-(1<<i)+1);
            if(i)For(j,n-(1<<i)+1)st[i][j]=merge(st[i-1][j],st[i-1][j+(1<<i-1)]);
            else For(i,n)st[0][pos[i]]=a[i];
        }
    }
    inline T query(int l,int r){
        int n=st[0].size();
        assert(0<=l),assert(l<=r),assert(r<n);
        int k=__lg(r-l+1);
        return merge(st[k][l],st[k][r-(1<<k)+1]);
    }
};
```

custom_hash.hpp

```cpp
struct custom_hash {
    static uint64_t splitmix64(uint64_t x){
        x+=0x9e3779b97f4a7c15;
        x=(x^(x>>30))*0xbf58476d1ce4e5b9;
        x=(x^(x>>27))*0x94d049bb133111eb;
        return x^(x>>31);
    }
    size_t operator()(uint64_t x)const{
```

```cpp
        static const uint64_t
        ↪   FIXED_RANDOM=chrono::steady_clock::now().time_since_epoch().count();
        return splitmix64(x+FIXED_RANDOM);
    }
};
```

dq.hpp

```cpp
template<class T>
struct dq{
    int hd;
    V<T>q;
    inline dq(){hd=0;}
    inline T front(int k=0){assert(hd+k<q.size());return q[hd+k];}
    inline T back(int k=0){assert(hd+k<q.size());return q[q.size()-1-k];}
    inline int size(){return q.size()-hd;}
    inline void clear(){hd=0,V<T>().swap(q);}
    inline void push(const T &v){q.pb(v);}
    inline void pop_back(){q.qb();}
    inline void pop_front(){assert(hd<q.size());++hd;}
};
```

fraction.hpp

```cpp
struct fraction{
    ll p,q;
    inline void simplify(){ll g=gcd(p<0?-p:p,q);p/=g;q/=g;}
    inline explicit fraction(ll _p=0):p(_p),q(1){}
    inline fraction(ll _p,ll
    ↪   _q):p(_p),q(_q){assert(q);if(q<0)p=-p,q=-q;simplify();}
    inline explicit fraction(const string&s){size_t pos=s.find('.');q=1;if(pos==⌋
    ↪   string::npos)p=stoll(s);else{if(pos+1<s.size()){for(int
    ↪   i=0;i<s.size()-1-pos;i++)q*=10;p=(pos?stoll(s.substr(0,pos))*q:0)+stoll(s⌋
    ↪   .substr(pos+1));}else
    ↪   p=stoll(s.substr(0,pos));simplify();}}
    inline explicit fraction(const V<char>&s):fraction(string(s.begin(),s.end())){}
    inline fraction& operator=(const fraction&r){p=r.p;q=r.q;return *this;}
    inline fraction& operator=(ll r){p=r;q=1;return *this;}
    inline fraction operator+(const fraction&r)const{if(q==r.q)return{p+r.p,q};ll
    ↪   g=gcd(q,r.q),m=q/g;return{p*(r.q/g)+r.p*m,m*r.q};}
    inline fraction operator+(ll r)const{return{p+r*q,q};}
    inline fraction add(const fraction&r)const{return{p*r.q+r.p*q,q*r.q};}
    inline fraction operator-(const fraction&r)const{if(q==r.q)return{p-r.p,q};ll
    ↪   g=gcd(q,r.q),m=q/g;return{p*(r.q/g)-r.p*m,m*r.q};}
    inline fraction operator-(ll r)const{return{p-r*q,q};}
    inline fraction sub(const fraction&r)const{return{p*r.q-r.p*q,q*r.q};}
    inline fraction operator*(const fraction&r)const{fraction t;ll g1=gcd(p,r.q),⌋
    ↪   g2=gcd(r.p,q);t.p=(p/g1)*(r.p/g2);t.q=(q/g2)*(r.q/g1);return
    ↪   t;}
    inline fraction operator*(ll r)const{fraction t=*this;ll
    ↪   g=gcd(r,q);t.p*=r/g;t.q/=g;return t;}
    inline fraction mul(const fraction&r)const{return{p*r.p,q*r.q};}
```

```
    inline fraction operator/(const fraction&r)const{assert(r.p);fraction t;ll g1↵
    ↪   =gcd(p,r.p),g2=gcd(r.q,q);t.p=(p/g1)*(r.q/g2);t.q=(q/g2)*(r.p/g1);return
    ↪   t;}
    inline fraction operator/(ll r)const{assert(r);fraction t=*this;ll
    ↪   g=gcd(p,r);t.p/=g;t.q*=r/g;return t;}
    inline fraction div(const fraction&r)const{assert(r.p);return{p*r.q,q*r.p};}
    inline bool operator==(const fraction&r)const{return p==r.p&&q==r.q;}
    inline bool operator==(ll r)const{return p==r&&q==1;}
    inline bool eq(const fraction&r)const{return p==r.p&&q==r.q;}
    inline bool operator<(const fraction&r)const{return p*r.q<r.p*q;}
    inline bool operator<(ll r)const{ll g=gcd(p,r);return p/g<q*(r/g);}
    inline bool lt(const fraction&r)const{return p*r.q<r.p*q;}
    inline bool operator>(const fraction&r)const{return p*r.q>r.p*q;}
    inline bool operator>(ll r)const{ll g=gcd(p,r);return p/g>q*(r/g);}
    inline bool gt(const fraction&r)const{return p*r.q>r.p*q;}
    inline bool operator<=(const fraction&r)const{return p*r.q<=r.p*q;}
    inline bool operator<=(ll r)const{ll g=gcd(p,r);return p/g<=q*(r/g);}
    inline bool le(const fraction&r)const{return p*r.q<=r.p*q;}
    inline bool operator>=(const fraction&r)const{return p*r.q>=r.p*q;}
    inline bool operator>=(ll r)const{ll g=gcd(p,r);return p/g>=q*(r/g);}
    inline bool ge(const fraction&r)const{return p*r.q>=r.p*q;}
    inline string to_string()const{return ::to_string(p)+'/'+::to_string(q);}
};
```

modint.hpp

```
template<int p>
struct modint{
    int val;
    inline modint(int v=0):val(v){}
    inline modint& operator=(int v){val=v;return *this;}
    inline modint& operator+=(const
    ↪   modint&k){val=val+k.val>=p?val+k.val-p:val+k.val;return *this;}
    inline modint& operator-=(const
    ↪   modint&k){val=val-k.val<0?val-k.val+p:val-k.val;return *this;}
    inline modint& operator*=(const modint&k){val=int(1ll*val*k.val%p);return
    ↪   *this;}
    inline modint& operator^=(int k){modint
    ↪   r(1),b=*this;for(;k;k>>=1,b*=b)if(k&1)r*=b;val=r.val;return *this;}
    inline modint& operator/=(modint k){return *this*=(k^=p-2);}
    inline modint& operator+=(int k){val=val+k>=p?val+k-p:val+k;return *this;}
    inline modint& operator-=(int k){val=val<k?val-k+p:val-k;return *this;}
    inline modint& operator*=(int k){val=int(1ll*val*k%p);return *this;}
    inline modint& operator/=(int k){return *this*=((modint(k))^=p-2);}
    template<class T>friend modint operator+(modint a,T b){return a+=b;}
    template<class T>friend modint operator-(modint a,T b){return a-=b;}
    template<class T>friend modint operator*(modint a,T b){return a*=b;}
    template<class T>friend modint operator/(modint a,T b){return a/=b;}
    friend modint operator^(modint a,int b){return a^=b;}
    friend bool operator==(modint a,int b){return a.val==b;}
    friend bool operator!=(modint a,int b){return a.val!=b;}
    inline bool operator!()const{return !val;}
    inline modint operator-()const{return val?modint(p-val):modint(0);}
    inline modint operator++(int){modint t=*this;*this+=1;return t;}
```

```cpp
    inline modint& operator++(){return *this+=1;}
    inline modint operator--(int){modint t=*this;*this-=1;return t;}
    inline modint& operator--(){return *this-=1;}
};
using mi=modint<mod>;
```

pbds.hpp

```cpp
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;

template<class T>
struct rbt{
    typedef pair<T,int> pti;
    int cnt;
    typedef tree<pti,null_type,less<pti>,rb_tree_tag,␂
      ↪  tree_order_statistics_node_update>
      ↪  rbt_t;
    rbt_t t;
    inline rbt(){cnt=0;}
    inline void clear(){cnt=0,rbt_t().swap(t);}
    inline typename rbt_t::iterator begin(){return t.begin();}
    inline typename rbt_t::iterator end(){return t.end();}
    inline void insert(const T &x){t.insert({x,cnt++});}
    inline typename rbt_t::iterator find(const T &x){return t.lower_bound({x,0});}
    inline void erase(const T &x){t.erase(find(x));}
    inline T pre(const T &x){
        auto it=find(x);
        assert(it!=begin());
        return prev(it)->fi;
    }
    inline T nxt(const T &x){
        auto it=find(x+1);
        assert(it!=end());
        return it->fi;
    }
    // all 0-indexed
    inline int rk(const T &x){return t.order_of_key({x,0});}
    inline T at(unsigned x){return t.find_by_order(x)->fi;}
};

#include <ext/pb_ds/priority_queue.hpp>
inline V<ll> dijkstra(int n,int s,const V<V<pii>> &to){
    assert(0<=n),assert(0<=s),assert(s<n),assert(to.size()<=n);
    for(const V<pii> &i:to)
        for(const pii &j:i)
            assert(0<=min(j.fi,j.se)),assert(j.fi<n);
    V<ll>dis(n,infl);
    dis[s]=0;
    __gnu_pbds::priority_queue<pli,greater<pli>,pairing_heap_tag>q;
    V<decltype(q)::point_iterator>it(n);
    it[s]=q.push({0,s});
    while(q.size()){
```

```cpp
        int p=q.top().se;q.pop();
        for(const pii &i:to[p])
            if(ckmin(dis[i.fi],dis[p]+i.se)){
                if(it[i.fi]!=NULL)q.modify(it[i.fi],{dis[i.fi],i.fi});
                else it[i.fi]=q.push({dis[i.fi],i.fi});
            }
    }
    for(ll &i:dis)if(i==infl)i=-1;
    return dis;
}
```

poly.hpp

```cpp
inline V<mi> poly_conv_add(const V<mi> &_a,const V<mi> &_b,int g){ //
↪   c[k]=∑(a[i]*b[j]) for i+j=k verified with lg3803
    assert(_a.size()&&_b.size());
    if(max(_a.size(),_b.size())<17){
        V<mi>c(_a.size()+_b.size()-1);
        For(i,_a.size())For(j,_b.size())c[i+j]+=_a[i]*_b[j];
        return c;
    }
    int lg=0,n=1;
    while(n<_a.size()+_b.size()-1)++lg,n<<=1;
    V<mi>a=_a,b=_b;
    a.resize(n),b.resize(n);
    static V<V<int>>btf;
    while(btf.size()<=lg){
        int n=1<<btf.size();
        btf.pb({});
        V<int>&bf=btf.back();
        bf.resize(n);
        For(i,n)bf[i]=(bf[i>>1]>>1)|((i&1)?n>>1:0);
    }
    const V<int>&bf=btf[lg];
    auto NTT=[&](V<mi> &f,mi coef){
        For(i,n)if(i<bf[i])swap(f[i],f[bf[i]]);
        for(int k=1,l=2;k<n;k<<=1,l<<=1){
            mi wn=coef^((mod-1)/l);
            for(int i=0;i<n;i+=l){
                mi w=1;
                For(j,k){
                    mi x=f[i|j],y=w*f[i|j|k];
                    f[i|j]=x+y,f[i|j|k]=x-y;
                    w*=wn;
                }
            }
        }
    };
    NTT(a,g),NTT(b,g);
    For(i,n)a[i]*=b[i];
    NTT(a,mi(1)/g);
    a.resize(_a.size()+_b.size()-1);
    mi invn=mi(1)/n;
    for(mi &i:a)i*=invn;
```

```cpp
        return a;
}

inline V<mi> poly_conv_sub(const V<mi> &_a,const V<mi> &_b,int g){ //
↪   c[k]=∑(a[i]*b[j]) for i-j=k verified with gym105386H
        assert(_a.size()&&_b.size());
        V<mi>b=_b;
        reverse(ALL(b));
        b=poly_conv_add(_a,b,g);
        // (-b.size(),a.size()) -> [0,a.size())
        b.erase(b.begin(),b.begin()+_b.size()-1);
        return b;
}

inline int find_g(int m){
        auto phi=[&](int k){
                int ret=k;
                for(int i=2;i*i<=k;++i)if(k%i==0){ret-=ret/i;do k/=i;while(k%i==0);}
                if(k>1)ret-=ret/k;
                return ret;
        };
        int p=phi(m);
        V<int>fac;
        {
                int j=p;
                for(int i=2;i*i<=j;++i)if(j%i==0){fac.pb(p/i);do j/=i;while(j%i==0);}
                if(j>1)fac.pb(p/j);
        }
        auto check_g=[&](int g){
                auto qpow=[&](int x,int y){
                        int z=1;
                        for(;y;x=1ll*x*x%m,y>>=1)if(y&1)z=1ll*z*x%m;
                        return z;
                };
                if(qpow(g,p)!=1)return false;
                for(int i:fac)if(qpow(g,i)==1)return false;
                return true;
        };
        FOR(i,1,m)if(check_g(i))return i;
        return -1;
}
inline V<mi> poly_conv_mul(const V<mi> &_a,const V<mi> &_b,int g,int p,int pg=-1){
↪   // c[k]=∑(a[i]*b[j]) for i*j%p=k verified by qoj9247
        assert(_a.size()&&_b.size());
        if(!~pg)pg=find_g(p);
        assert(~pg);
        V<int>exp(p-1),lg(p);
        lg[0]=-1;
        for(int i=1,j=0;j<p-1;i=1ll*i*pg%p,++j)exp[j]=i,lg[i]=j;
        V<mi>a(p-1),b(p-1);
        FOR(i,1,_a.size())a[lg[i]]=_a[i];
        FOR(i,1,_b.size())b[lg[i]]=_b[i];
        V<mi>c=poly_conv_add(a,b,g);
        FOR(i,p-1,c.size())c[i-(p-1)]+=c[i];
```

```cpp
        V<mi>d(p);
        d[0]=_a[0]*reduce(ALL(_b))+_b[0]*reduce(ALL(_a))-_a[0]*_b[0];
        For(i,p-1)d[exp[i]]=c[i];
        return d;
}

inline V<mi> poly_conv_div(const V<mi> &_a,const V<mi> &_b,int g,int p,int pg=-1){
↳   // c[k]=∑(a[i]*b[j]) for i/j%p=k not verified
        assert(_a.size()&&_b.size()),assert(!_b[0].val);
        V<int>inv(p);
        inv[1]=1;
        FOR(i,1,p)inv[i]=1ll*(p-p/i)*inv[p%i]%mod;
        V<mi>b(p);
        FOR(i,1,_b.size())b[inv[i]]=_b[i];
        return poly_conv_mul(_a,b,g,p,pg);
}

inline V<mi> poly_conv_and(const V<mi> &_a,const V<mi> &_b){ // c[k]=∑(a[i]*b[j])
↳   for i&j=k verified with lg4717
        assert(_a.size()&&_b.size());
        int n=1;
        while(n<max(_a.size(),_b.size()))n<<=1;
        V<mi>a=_a,b=_b;
        a.resize(n),b.resize(n);
        auto FWT=[&](V<mi> &f,int coef){
            for(int k=1,l=2;k<n;k<<=1,l<<=1)for(int
            ↳  i=0;i<n;i+=l)For(j,k)f[i|j]+=f[i|j|k]*coef;
        };
        FWT(a,1),FWT(b,1);
        For(i,n)a[i]*=b[i];
        FWT(a,mod-1);
        return a;
}

inline V<mi> poly_conv_or(const V<mi> &_a,const V<mi> &_b){ // c[k]=∑(a[i]*b[j])
↳   for i|j=k verified with lg4717
        assert(_a.size()&&_b.size());
        int n=1;
        while(n<max(_a.size(),_b.size()))n<<=1;
        V<mi>a=_a,b=_b;
        a.resize(n),b.resize(n);
        auto FWT=[&](V<mi> &f,int coef){
            for(int k=1,l=2;k<n;k<<=1,l<<=1)for(int
            ↳  i=0;i<n;i+=l)For(j,k)f[i|j|k]+=f[i|j]*coef;
        };
        FWT(a,1),FWT(b,1);
        For(i,n)a[i]*=b[i];
        FWT(a,mod-1);
        return a;
}

inline V<mi> poly_conv_xor(const V<mi> &_a,const V<mi> &_b){ // c[k]=∑(a[i]*b[j])
↳   for i^j=k verified with lg4717
        assert(_a.size()&&_b.size());
```

```cpp
    int n=1;
    while(n<max(_a.size(),_b.size())))n<<=1;
    V<mi>a=_a,b=_b;
    a.resize(n),b.resize(n);
    auto FWT=[&](V<mi> &f,int coef){
        for(int k=1,l=2;k<n;k<<=1,l<<=1)for(int i=0;i<n;i+=l)For(j,k){
            mi x=f[i|j],y=f[i|j|k];
            f[i|j]=(x+y)*coef,f[i|j|k]=(x-y)*coef;
        }
    };
    FWT(a,1),FWT(b,1);
    For(i,n)a[i]*=b[i];
    FWT(a,mod+1>>1);
    return a;
}


inline V<mi> poly_conv_gcd(const V<mi> &_a,const V<mi> &_b){ // c[k]=∑(a[i]*b[j])
 ↪   for gcd(i,j)=k verified with lc418t4
    assert(_a.size()&&_b.size());
    int n=max(_a.size(),_b.size());
    V<mi>a=_a,b=_b;
    a.resize(n),b.resize(n);
    V<int>pri;
    V<bool>vis(n);
    FOR(i,2,n)if(!vis[i]){
        pri.pb(i);
        for(int k=(n-1)/i,j=k*i;k;j-=i,--k)a[k]+=a[j],b[k]+=b[j],vis[j]=true;
    }
    FOR(i,1,n)a[i]*=b[i];
    for(int i:pri)for(int j=i,k=1;j<n;j+=i,++k)a[k]-=a[j];
    a[0]=_a[0]*_b[0];
    FOR(i,1,n)a[i]+=_a[0]*_b[i]+_b[0]*_a[i];
    return a;
}

inline V<mi> poly_conv_lcm(const V<mi> &_a,const V<mi> &_b){ // c[k]=∑(a[i]*b[j])
 ↪   for lcm(i,j)=k not verified
    assert(_a.size()&&_b.size());
    int n=max(_a.size(),_b.size());
    V<mi>a=_a,b=_b;
    a.resize(n),b.resize(n);
    V<int>pri;
    V<bool>vis(n);
    FOR(i,2,n)if(!vis[i]){
        pri.pb(i);
        for(int j=i,k=1;j<n;j+=i,++k)a[j]+=a[k],b[j]+=b[k],vis[j]=true;
    }
    FOR(i,1,n)a[i]*=b[i];
    for(int i:pri)for(int k=(n-1)/i,j=k*i;k;j-=i,--k)a[j]-=a[k];
    a[0]=_a[0]*_b[0];
    FOR(i,1,n)a[i]+=_a[0]*_b[i]+_b[0]*_a[i];
    return a;
```

```
}

inline V<mi> poly_inv(const V<mi> &a,int g){ // b=1/a verified with lg4238
    assert(a.size()),assert(a[0].val);
    V<mi>b{1/a[0]};
    mi invg=mi(1)/g,invm=1;
    int m=1;
    while(b.size()<a.size()){
        int n=min(a.size(),b.size()<<1);
        while(m<=n-1<<1)invm*=mod+1>>1,m<<=1;
        V<mi>c(a.begin(),a.begin()+n);
        b.resize(m),c.resize(m);
        V<int>bf(m);
        For(i,m)bf[i]=(bf[i>>1]>>1)|((i&1)?m>>1:0);
        auto NTT=[&](V<mi> &f,mi coef){
            For(i,m)if(i<bf[i])swap(f[i],f[bf[i]]);
            for(int k=1,l=2;k<m;k<<=1,l<<=1){
                mi wn=coef^((mod-1)/l);
                for(int i=0;i<m;i+=l){
                    mi w=1;
                    For(j,k){
                        mi x=f[i|j],y=w*f[i|j|k];
                        f[i|j]=x+y,f[i|j|k]=x-y;
                        w*=wn;
                    }
                }
            }
        };
        NTT(b,g),NTT(c,g);
        For(i,m)b[i]*=2-b[i]*c[i];
        NTT(b,invg);
        b.resize(n);
        for(mi &i:b)i*=invm;
    }
    return b;
}

inline V<mi> poly_diff(const V<mi> &a){ // b=a'
    int n=a.size();
    assert(n);
    if(n==1)return {0};
    V<mi>b(n-1);
    For(i,n-1)b[i]=a[i+1]*(i+1);
    return b;
}

inline V<mi> poly_intg(const V<mi> &a){ // b=∫a
    int n=a.size();
    assert(n);
    V<mi>b(n+1),inv(n+1);
    b[1]=a[0],inv[1]=1;
    FOR(i,2,n)b[i]=a[i-1]*(inv[i]=(mod-mod/i)*inv[mod%i]);
    return b;
}
```

```cpp
inline V<mi> poly_ln(const V<mi> &a,int g){ // b=ln(a) verified with lg4725
    int n=a.size();
    assert(n),assert(a[0].val==1);
    V<mi>b=poly_conv_add(poly_diff(a),poly_inv(a,g),g);
    b.resize(n);
    return poly_intg(b);
}

inline V<mi> poly_exp(const V<mi> &a,int g){ // b=exp(a) verified with lg4726
    int n=a.size();
    assert(n);
    V<mi>b{1};
    if(a[0].val){
        mi e=0,ifac=mod-1;
        Rep(i,mod)e+=ifac,ifac*=i;
        b[0]=e^a[0].val; // check that a[0] isnt modulo
    }
    while(b.size()<a.size()){
        int m=min(b.size()<<1,a.size());
        b.resize(m);
        V<mi>c=poly_ln(b,g);
        For(i,m)c[i]=a[i]-c[i];
        ++c[0];
        b=poly_conv_add(b,c,g);
        b.resize(m);
    }
    return b;
}

inline V<mi> poly_series(const V<mi> &a,mi b0,int g){ // b[i]=∑(b[j]*a[i-j]) for
 ↪  j>0 verified with lg4721
    assert(a.size());
    V<mi>b=a;
    b[0]=1;
    FOR(i,1,b.size())b[i]=-b[i];
    b=poly_inv(b,g);
    if(b0.val!=1)for(mi &i:b)i*=b0;
    return b;
}

inline V<mi> poly_pow(const V<mi> &_a,mi b,int g){ // c=a^(b%mod) verified with
 ↪  lg5245
    int n=_a.size();
    assert(n);
    V<mi>a(n);
    if(!b){
        a[0]=1;
        return a;
    }
    int i=0;
    while(i<n&&!_a[i])++i;
    if(i==n)return a;
    ll z=1ll*b.val*i;
```

```cpp
        if(z>=n)return a;
        assert(_a[i].val==1);
        a=poly_ln(V<mi>(_a.begin()+i,_a.end()),g);
        for(mi &j:a)j*=b;
        a=poly_exp(a,g);
        V<mi>ret(z);
        ret.insert(ret.end(),a.begin(),a.begin()+n-z);
        return ret;
}

inline V<mi> poly_pow(const V<mi> &_a,ll b,int g){ // c=a^b verified with Library
    ↪ Checker
        int n=_a.size();
        assert(n);
        V<mi>a(n);
        if(!b){
            a[0]=1;
            return a;
        }
        int i=0;
        while(i<n&&!_a[i])++i;
        if(i==n||__int128(b)*i>=n)return a;
        a=V<mi>(_a.begin()+i,_a.end());
        mi coef=a[0],inv=1/coef;
        for(mi &j:a)j*=inv;
        a=poly_ln(a,g);
        mi _b=b%mod;
        for(mi &j:a)j*=_b;
        a=poly_exp(a,g);
        coef^=b%(mod-1);
        for(mi &j:a)j*=coef;
        ll z=b*i;
        V<mi>ret(z);
        ret.insert(ret.end(),a.begin(),a.begin()+n-z);
        return ret;
}

inline V<mi> poly_multi_pt(const V<mi> &_a,const V<mi> &b,int g){ // c[i]=a(b[i])
    ↪ verified with lg5050
        assert(_a.size());
        if(b.empty())return {};
        int n=max(_a.size(),b.size());
        V<V<mi>>t(n<<2);
        auto build=[&](auto &&self,int p,int l,int r)->void{
            if(l==r){
                t[p]={1,l<b.size()?-b[r]:0};
                return;
            }
            int mid=l+r>>1;
            self(self,p<<1,l,mid);
            self(self,p<<1|1,mid+1,r);
            t[p]=poly_conv_add(t[p<<1],t[p<<1|1],g);
        };
        build(build,1,0,n-1);
```

```
    auto poly_conv_sub=[&](const V<mi> &_a,const V<mi> &_b,int g){
        assert(_b.size()),assert(_a.size()>=_b.size());
        V<mi>b=_b;
        reverse(ALL(b));
        b=poly_conv_add(_a,b,g);
        return V<mi>(b.begin()+_b.size()-1,b.end());
    };
    V<mi>ret(b.size());
    auto push_down=[&](auto &&self,int p,int l,int r,V<mi> c)->void{
        if(l>=b.size())return;
        if(l==r){
            ret[l]=c[0];
            return;
        }
        c.resize(r-l+1);
        int mid=l+r>>1;
        self(self,p<<1,l,mid,poly_conv_sub(c,t[p<<1|1],g));
        self(self,p<<1|1,mid+1,r,poly_conv_sub(c,t[p<<1],g));
    };
    V<mi>a=_a;
    a.resize(n+1);
    push_down(push_down,1,0,n-1,poly_conv_sub(a,poly_inv(t[1],g),g));
    return ret;
}

inline V<mi> poly_prod(const V<V<mi>> &a,int g){ // b=∏(a[i])
    assert(a.size());
    auto cmp=[&](const V<mi> &x,const V<mi> &y){return x.size()>y.size();};
    priority_queue<V<mi>,V<V<mi>>,decltype(cmp)>q(cmp);
    for(const auto &i:a)q.push(i);
    while(q.size()>1){
        V<mi>x=q.top();q.pop();
        V<mi>y=q.top();q.pop();
        q.push(poly_conv_add(x,y,g));
    }
    return q.top();
}

inline V<mi> poly_multi_pt_sum(const V<mi> &a,int m,int g){ // b[i]=sum(a[j]^i)
↪   for i in [0,m]
    int n=a.size();
    assert(n);
    V<V<mi>>b(max(n,m));
    For(i,max(n,m))b[i]={1,-a[i]};
    V<mi>c=poly_ln(poly_prod(b,g),g);
    c.resize(m+1);
    c[0]=n;
    FOR(i,1,m+1)c[i]*=mod-i;
    return c;
}
```

trie.hpp

```cpp
struct trie{
    int siz;
    trie *son[2];
    inline trie(){siz=0,son[0]=son[1]=NULL;}
};
void insert(int dep,trie *p,int k){
    ++p->siz;
    if(dep<0)return;
    int nxt=k>>dep&1;
    if(!p->son[nxt])p->son[nxt]=new trie();
    insert(dep-1,p->son[nxt],k);
}
int query(int dep,trie *p,int k,int lim){
    if(!p)return 0;
    if(dep<0)return p->siz;
    int nxt=k>>dep&1;
    if(lim>>dep&1)return
    ↪   (p->son[nxt]?p->son[nxt]->siz:0)+query(dep-1,p->son[nxt^1],k,lim);
    return query(dep-1,p->son[nxt],k,lim);
}
void insert(int dep,trie *p1,trie *p2,int k){
    if(p1)p2->siz=p1->siz;
    ++p2->siz;
    if(dep<0)return;
    int nxt=k>>dep&1;
    if(p1)p2->son[nxt^1]=p1->son[nxt^1];
    p2->son[nxt]=new trie();
    insert(dep-1,p1?p1->son[nxt]:NULL,p2->son[nxt],k);
}
int query(int dep,trie *p1,trie *p2,int k){
    if(dep<0)return 0;
    int nxt=k>>dep&1;
    if(p2->son[nxt^1]&&(!p1||!p1->son[nxt^1]||p2->son[nxt^1]->siz>p1->son[nxt^1]-
    ↪   >siz))return
    ↪   query(dep-1,p1?p1->son[nxt^1]:NULL,p2->son[nxt^1],k)|(1<<dep);
    return query(dep-1,p1?p1->son[nxt]:NULL,p2->son[nxt],k);
}
```

vector.hpp

```cpp
template<class T>
inline V<V<T>> rot(const V<V<T>>& v){
    V<V<T>>ret(v[0].size(),V<T>(v.size()));
    For(i,v.size())
        For(j,v[0].size())
            ret[j][v.size()-i-1]=v[i][j];
    return ret;
}

inline ll contor(const V<int> &v){
    int d=*min_element(ALL(v)),n=v.size();
    V<bool>vis(n);
```

```cpp
    for(int i:v)vis[i-d]=true;
    if(any_of(ALL(vis),[](bool b){return !b;}))return -1;
    V<ll>fac(n);
    fac[0]=1;
    BIT3<int>t(n);
    FOR(i,1,n+1){
        if(i<n)fac[i]=fac[i-1]*i;
        ++t.c[i];
        if(i+(i&-i)<=n)t.c[i+(i&-i)]+=t.c[i];
    }
    ll ret=0;
    For(i,n){
        t.add(v[i]-d,-1);
        ret+=fac[n-i-1]*t.query(v[i]-d);
    }
    return ret;
}
inline V<int> inv_contor(int n,ll k){
    V<ll>fac(n+1);
    fac[0]=1;
    FOR(i,1,n+1)fac[i]=fac[i-1]*i;
    if(k>=fac[n])return {-1};
    V<int>ret(n);
    V<bool>vis(n);
    For(i,n){
        int dgt=k/fac[n-i-1]+1,j=-1;
        k%=fac[n-i-1];
        do dgt-=!vis[++j];while(dgt);
        ret[i]=j,vis[j]=true;
    }
    return ret;
}
```

哈希.hpp

```cpp
template<int base=2333>
struct mhsh{
    // 0-indexed
    V<ull>bs,h;
    inline mhsh(){}
    inline mhsh(const string &s){
        bs.reserve(s.size()),h.reserve(s.size());
        bs.pb(1),h.pb(s[0]);
        FOR(i,1,s.size())bs.pb(bs.back()*base),h.pb(h.back()*base+s[i]);
    }
    inline mhsh(const V<int> &v){
        bs.reserve(v.size()),h.reserve(v.size());
        bs.pb(1),h.pb(v[0]);
        FOR(i,1,v.size())bs.pb(bs.back()*base),h.pb(h.back()*base+v[i]);
    }
    inline ull get(int l,int r){
        assert(0<=l),assert(l<=r),assert(r<h.size());
        return h[r]-(l?h[l-1]*bs[r-l+1]:0);
    }
```

```cpp
    inline int lcp(int x,int y){
        assert(0<=min(x,y)),assert(max(x,y)<h.size());
        int l=1,r=h.size()-max(x,y),ret=0;
        while(l<=r){
            int mid=l+r>>1;
            if(get(x,x+mid-1)==get(y,y+mid-1))l=mid+1,ret=mid;
            else r=mid-1;
        }
        return ret;
    }
};

template<int base=2337,int mod=998244853>
struct modhsh{
    // 0-indexed
    V<ull>bs,h;
    inline modhsh(){}
    inline modhsh(const string &s){
        bs.reserve(s.size()),h.reserve(s.size());
        bs.pb(1),h.pb(s[0]);
        FOR(i,1,s.size())bs.pb(bs.back()*base%mod),h.pb((h.back()*base+s[i])%mod);
    }
    inline modhsh(const V<int> &v){
        bs.reserve(v.size()),h.reserve(v.size());
        bs.pb(1),h.pb(v[0]);
        FOR(i,1,v.size())bs.pb(bs.back()*base%mod),h.pb((h.back()*base+v[i])%mod);
    }
    inline ull get(int l,int r){
        assert(0<=l),assert(l<=r),assert(r<h.size());
        ull ret=h[r]+mod-(l?h[l-1]*bs[r-l+1]%mod:0);
        return ret>=mod?ret-mod:ret;
    }
};
template<int base1=2337,int mod1=998244853,int base2=2333,int mod2=1'000'000'009>
struct dmhsh{
    // 0-indexed
    modhsh<base1,mod1>hsh1;
    modhsh<base2,mod2>hsh2;
    inline dmhsh(const string &s){
        hsh1=modhsh<base1,mod1>(s),hsh2=modhsh<base2,mod2>(s);
    }
    inline dmhsh(const V<int> &v){
        hsh1=modhsh<base1,mod1>(v),hsh2=modhsh<base2,mod2>(v);
    }
    inline pair<ull,ull> get(int l,int r){
        assert(0<=l),assert(l<=r),assert(r<hsh1.h.size());
        return {hsh1.get(l,r),hsh2.get(l,r)};
    }
    inline int lcp(int x,int y){
        assert(0<=min(x,y)),assert(max(x,y)<hsh2.h.size());
        int l=1,r=hsh2.h.size()-max(x,y),ret=0;
        while(l<=r){
            int mid=l+r>>1;
            if(get(x,x+mid-1)==get(y,y+mid-1))l=mid+1,ret=mid;
```

```
                else r=mid-1;
            }
            return ret;
        }
};

mt19937 rnd(time(0));
inline int genPri(int l,int r){
    auto isp=[&](int k){
        if(k<2)return false;
        for(int i=2;i*i<=k;++i)if(k%i==0)return false;
        return true;
    };
    int p=uniform_int_distribution<int>(l,r)(rnd);
    while(!isp(p))++p;
    return p;
};
struct rndhsh{
    // 0-indexed
    int base,mod;
    V<ull>bs,h;
    inline rndhsh(){base=genPri(2,1e5),mod=genPri(2,1e9);}
    inline rndhsh(const string &s){
        bs.reserve(s.size()),h.reserve(s.size());
        bs.pb(1),h.pb(s[0]);
        FOR(i,1,s.size())bs.pb(bs.back()*base%mod),h.pb((h.back()*base+s[i])%mod);
    }
    inline rndhsh(const V<int> &v){
        bs.reserve(v.size()),h.reserve(v.size());
        bs.pb(1),h.pb(v[0]);
        FOR(i,1,v.size())bs.pb(bs.back()*base%mod),h.pb((h.back()*base+v[i])%mod);
    }
    inline ull get(int l,int r){
        assert(0<=l),assert(l<=r),assert(r<h.size());
        ull ret=h[r]+mod-(l?h[l-1]*bs[r-l+1]%mod:0);
        return ret>=mod?ret-mod:ret;
    }
};
struct drhsh{
    // 0-indexed
    rndhsh hsh1;
    rndhsh hsh2;
    inline drhsh(const string &s){
        hsh1=rndhsh(s),hsh2=rndhsh(s);
    }
    inline drhsh(const V<int> &v){
        hsh1=rndhsh(v),hsh2=rndhsh(v);
    }
    inline pair<ull,ull> get(int l,int r){
        assert(0<=l),assert(l<=r),assert(r<hsh1.h.size());
        return {hsh1.get(l,r),hsh2.get(l,r)};
    }
    inline int lcp(int x,int y){
        assert(0<=min(x,y)),assert(max(x,y)<hsh2.h.size());
```

```
        int l=1,r=hsh2.h.size()-max(x,y),ret=0;
        while(l<=r){
            int mid=l+r>>1;
            if(get(x,x+mid-1)==get(y,y+mid-1))l=mid+1,ret=mid;
            else r=mid-1;
        }
        return ret;
    }
};
```

图论.hpp

```
inline V<ll> bfs01(int n,int s,const V<V<pii>> &to){
    assert(0<=n),assert(0<=s),assert(s<n),assert(to.size()<=n);
    for(const V<pii> &i:to)
        for(const pii &j:i)
            assert(0<=min(j.fi,j.se)),assert(j.fi<n);
    V<ll>dis(n,infl);
    dis[s]=0;
    deque<int>q;
    q.pb(s);
    V<bool>vis(n); // added vis to prevent an obvious error
    while(q.size()){
        int p=q.front();q.qf();
        if(vis[p])continue;
        vis[p]=true;
        for(const pii
        ↪ &i:to[p])if(ckmin(dis[i.fi],dis[p]+i.se))i.se?q.pb(i.fi):q.pf(i.fi);
    }
    for(ll &i:dis)if(i==infl)i=-1;
    return dis;
}


template<class T>
inline V<ll> dijkstra(int n,int s,const V<V<pair<int,T>>> &to,ll null=-1){
    V<ll>dis(n,infl);
    dis[s]=0;
    typedef pair<int,ll> pil;
    auto cmp=[&](const pil &x,const pil &y){return x.se>y.se;};
    priority_queue<pil,V<pil>,decltype(cmp)>q(cmp);
    q.emplace(s,0);
    V<bool>vis(n);
    while(q.size()){
        int p=q.top().fi;q.pop();
        if(vis[p])continue;
        vis[p]=true;
        for(const auto
        ↪ &[i,j]:to[p])if(ckmin(dis[i],dis[p]+j)&&!vis[i])q.emplace(i,dis[i]);
    }
    for(ll &i:dis)if(i==infl)i=null;
    return dis;
}
```

```cpp
inline V<array<int,3>> kruskal(int n,const V<V<pii>> &to,function<bool(const
↪   array<int,3> &,const array<int,3> &)>cmp=[](const array<int,3> x,const
↪   array<int,3> &y){return x[2]<y[2];}){
    assert(0<=n),assert(to.size()<=n);
    for(const V<pii> &i:to)for(const pii &j:i)assert(j.fi<n);
    V<array<int,3>>e;
    For(i,to.size())for(const pii
↪   &j:to[i])assert(0<=j.fi),assert(j.fi<n),e.pb({i,j.fi,j.se});
    sort(ALL(e),cmp);
    dsu d(n);
    V<array<int,3>>ret;
    for(auto &i:e)if(d.merge(i[0],i[1]))ret.pb(i);
    return ret;
}

struct ring{
    int clr;
    V<int>id;
    V<V<int>>scc,to;
    inline void init(const V<V<int>>&to){
        int cnt=clr=0,n=to.size();
        V<bool>cur(n);
        V<int>dfn(n),low(n);
        V<int>(n,-1).swap(id),V<V<int>>().swap(scc);
        stack<int>st;
        function<void(int)>tarjan=[&](int p){
            cur[p]=true;
            dfn[p]=low[p]=++cnt;
            st.push(p);
            for(int i:to[p]){
                assert(0<=i&&i<n);
                if(!dfn[i])tarjan(i),ckmin(low[p],low[i]);
                else if(cur[i])ckmin(low[p],dfn[i]);
            }
            if(dfn[p]==low[p]){
                scc.pb(V<int>());
                int k;
                do{
                    k=st.top();st.pop();
                    cur[k]=false,id[k]=clr,scc[clr].pb(k);
                }while(k!=p);
                ++clr;
            }
        };
        For(i,n)if(!dfn[i])tarjan(i);
        V<int>lst(clr,-1);
        V<V<int>>(clr).swap(this->to);
        For(i,clr){
            lst[i]=i;
            for(int j:scc[i])for(int
↪           k:to[j])if(lst[id[k]]!=i)lst[id[k]]=i,this->to[i].pb(id[k]);
        }
    }
    inline ring(const V<V<int>>&to){init(to);}
```

19

```cpp
        inline ring(){}
};

struct vDCC{
    int clr;
    V<bool>cut;
    V<V<int>>dcc,to;
    inline void init(const V<V<int>>&to){
        int cnt=0,n=clr=to.size();
        V<int>dfn(n),low(n);
        V<bool>(n).swap(cut),V<V<int>>().swap(dcc);
        V<V<int>>(n).swap(this->to);
        For(i,n)
            if(!dfn[i]){
                stack<int>st;
                function<void(int,int)>tarjan=[&](int p,int fa){
                    dfn[p]=low[p]=++cnt;
                    int flag_son=0;
                    st.push(p);
                    for(int i:to[p]){
                        assert(0<=i&&i<n);
                        if(!dfn[i]){
                            tarjan(i,p),ckmin(low[p],low[i]);
                            if(low[i]>=dfn[p]){
                                if(fa!=-1||flag_son++)cut[p]=true;
                                this->dcc.pb(V<int>()),this->to.pb(V<int>());
                                int k;
                                do{
                                    k=st.top();st.pop();
                                    this->dcc.back().pb(k);
                                    this->to[k].pb(clr),this->to[clr].pb(k);
                                }while(k!=i);
                                this->dcc.back().pb(p);
                                this->to[p].pb(clr),this->to[clr++].pb(p);
                            }
                        }
                        else ckmin(low[p],dfn[i]);
                    }
                    if(!~fa&&!flag_son)this->dcc.pb({p});
                };
                tarjan(i,-1);
            }
    }
    inline vDCC(const V<V<int>>&to){init(to);}
    inline vDCC(){}
};

struct eDCC{
    int clr;
    V<V<int>>dcc,to;
    V<int>id;
    inline void init(const V<V<int>>&to){
        int cnt=clr=0,n=to.size();
        V<int>dfn(n),low(n);
```

```cpp
        V<V<int>>().swap(dcc),V<int>(n,-1).swap(id);
        stack<int>st;
        function<void(int,int)>tarjan=[&](int p,int fa){
            dfn[p]=low[p]=++cnt;
            bool flag=false;
            st.push(p);
            for(int i:to[p]){
                if(i!=fa){
                    if(!dfn[i])tarjan(i,p),ckmin(low[p],low[i]);
                    else ckmin(low[p],dfn[i]);
                }
                if(i==fa){
                    if(flag)ckmin(low[p],dfn[i]);
                    else flag=true;
                }
            }
            if(dfn[p]<=low[p]){
                dcc.pb(V<int>());
                int k;
                do{
                    k=st.top();st.pop();
                    id[k]=clr,dcc[clr].pb(k);
                }while(k!=p);
                ++clr;
            }
        };
        For(i,n)if(!dfn[i])tarjan(i,-1);
        V<int>lst(clr,-1);
        V<V<int>>(clr).swap(this->to);
        For(i,clr){
            lst[i]=i;
            for(int j:dcc[i])for(int
            ↪  k:to[j])if(lst[id[k]]!=i)lst[id[k]]=i,this->to[i].pb(id[k]);
        }
    }
    inline eDCC(const V<V<int>>&to){init(to);}
    inline eDCC(){}
};

struct range_2sat{
    int n;
    V<V<int>>to;
    inline int idx(int l,int r){return (l+r|l!=r)>>1;}
    #define p idx(l,r)
    inline void resize(int n_){
        n=n_;
        V<V<int>>((n<<1)+(n-1<<2)).swap(to);
        function<int(int,int,int)>build_dw=[&](int l,int r,int k){
            if(l==r)return (k&1)*n+l;
            int mid=l+r>>1;
            to[(n<<1)+k*(n-1)+p].pb(build_dw(l,mid,k));
            to[(n<<1)+k*(n-1)+p].pb(build_dw(mid+1,r,k));
            return (n<<1)+k*(n-1)+p;
        };
```

```
        build_dw(0,n-1,0),build_dw(0,n-1,1);
        function<int(int,int,int)>build_up=[&](int l,int r,int k){
            if(l==r)return (k&1)*n+r;
            int mid=l+r>>1;
            to[build_up(l,mid,k)].pb((n<<1)+k*(n-1)+p);
            to[build_up(mid+1,r,k)].pb((n<<1)+k*(n-1)+p);
            return (n<<1)+k*(n-1)+p;
        };
        build_up(0,n-1,2),build_up(0,n-1,3);
    }
    inline range_2sat(){}
    inline range_2sat(int n_){resize(n_);}
    inline V<int> range_dw(int ql,int qr,int k){
        V<int>ret;
        function<void(int,int)>dfs=[&](int l,int r){
            if(ql<=l&&r<=qr){
                if(l==r)ret.pb(k*n+l);
                else ret.pb((n<<1)+k*(n-1)+p);
                return;
            }
            int mid=l+r>>1;
            if(ql<=mid)dfs(l,mid);
            if(qr>mid)dfs(mid+1,r);
        };
        dfs(0,n-1);
        return ret;
    }
    inline V<int> range_up(int ql,int qr,int k){
        V<int>ret;
        function<void(int,int)>dfs=[&](int l,int r){
            if(ql<=l&&r<=qr){
                if(l==r)ret.pb(k*n+r);
                else ret.pb((n<<1)+(k+2)*(n-1)+p);
                return;
            }
            int mid=l+r>>1;
            if(ql<=mid)dfs(l,mid);
            if(qr>mid)dfs(mid+1,r);
        };
        dfs(0,n-1);
        return ret;
    }
#undef p
    inline void link_pp(int x,int y,bool op_x,bool op_y,bool rev=true){
        to[op_x*n+x].pb(op_y*n+y);
        if(rev)to[(op_y^1)*n+y].pb((op_x^1)*n+x);
    }
    inline void link_pr(int x,int yl,int yr,bool op_x,bool op_y,bool rev=true){
        for(int y:range_dw(yl,yr,op_y))to[op_x*n+x].pb(y);
        if(rev)for(int y:range_up(yl,yr,op_y^1))to[y].pb((op_x^1)*n+x);
    }
    inline void link_rp(int xl,int xr,int y,bool op_x,bool op_y,bool rev=true){
        for(int x:range_up(xl,xr,op_x))to[x].pb(op_y*n+y);
        if(rev)for(int x:range_dw(xl,xr,op_x^1))to[(op_y^1)*n+y].pb(x);
```

```cpp
    }
    inline void link_rr(int xl,int xr,int yl,int yr,bool op_x,bool op_y,bool
    ↪   rev=true){
        V<int>X=range_up(xl,xr,op_x);
        for(int y:range_dw(yl,yr,op_y))for(int x:X)to[x].pb(y);
        if(rev){
            V<int>Y=range_up(yl,yr,op_y^1);
            for(int x:range_dw(xl,xr,op_x^1))for(int y:Y)to[y].pb(x);
        }
    }
};
```

堆.hpp

```cpp
template<class T,class U=less<T>>
struct delpq{
    priority_queue<T,V<T>,U>q1,q2;
    inline delpq(){}
    inline delpq(const U &func){priority_queue<T,V<T>,U>(func).swap(q1),
    ↪   priority_queue<T,V<T>,U>(func).swap(q2);}
    inline void push(const T &x){q1.push(x);}
    inline void pop(const T &x){q2.push(x);}
    inline T top(){
        while(q2.size()&&q1.top()==q2.top())q1.pop(),q2.pop();
        assert(q1.size());
        return q1.top();
    }
    inline bool empty(){return q1.size()==q2.size();}
    inline int size(){assert(q1.size()>=q2.size());return q1.size()-q2.size();}
};

template<class T>
struct kpq{
    int k;
    multiset<T>s1,s2;
    ll sum;
    inline kpq(int _k=0):k(_k),sum(0){}
    inline void insert(const T &x){
        if(s1.size()<k)s1.insert(x),sum+=x;
        else{
            if(x>*s1.begin())s2.insert(*s1.begin()),sum-=*s1.begin(),s1.erase(s1.
            ↪   begin()),s1.insert(x),sum+=x;
            else s2.insert(x);
        }
    }
    inline void erase(const T &x){
        if(s1.size()&&x<*s1.begin()){
            auto it=s2.find(x);
            assert(it!=s2.end());
            s2.erase(it);
        }
        else{
            auto it=s1.find(x);
```

```
            assert(it!=s1.end());
            s1.erase(it);
            sum-=x;
            if(s1.size()<k&&s2.size())s1.insert(*s2.rbegin()),sum+=*s2.rbegin(),⏎
                ↪  s2.erase(prev(s2.end())));
        }
    }
};
```

字符串.hpp

```
inline int lcs(const string &a,const string &b){
    if(a.empty()||b.empty())return 0;
    int n=a.size(),m=b.size(),k=(n+62)/63;
    V<ull>f(k);
    char mn=*min_element(ALL(a)),mx=*max_element(ALL(a));
    V<V<ull>>g(mx-mn+1,V<ull>(k));
    For(i,n)g[a[i]-mn][i/63]|=1ull<<i%63;
    for(char i:b){
        if(i<mn||i>mx)continue;
        i-=mn;
        ull z=1;
        For(j,k){
            ull x=f[j],y=f[j]|g[i][j];
            ((x<<=1)|=z)+=(~y)&((1ull<<63)-1);
            f[j]=x&y,z=x>>63;
        }
    }
    return accumulate(ALL(f),0,[&](int x,ull y){return
        ↪  x+__builtin_popcountll(y);});
}
template<class T>
inline int lcs(const V<T> &a,const V<T> &b){
    if(a.empty()||b.empty())return 0;
    int n=a.size(),m=b.size(),k=(n+62)/63;
    disc<T>d(a);
    V<ull>f(k);
    V<V<ull>>g(d.size(),V<ull>(k));
    For(i,n)g[d.query(a[i])][i/63]|=1ull<<i%63;
    for(const T &i:b){
        auto it=lower_bound(ALL(d.d),i);
        if(it==d.d.end()||*it!=i)continue;
        i=it-d.d.begin();
        ull z=1;
        For(j,k){
            ull x=f[j],y=f[j]|g[i][j];
            ((x<<=1)|=z)+=(~y)&((1ull<<63)-1);
            f[j]=x&y,z=x>>63;
        }
    }
    return accumulate(ALL(f),0,[&](int x,ull y){return
        ↪  x+__builtin_popcountll(y);});
}
```

```cpp
struct subseq_table{
    V<V<int>>nxt;
    inline subseq_table(const string &v){
        int n=v.size();
        V<V<int>>(128).swap(nxt);
        For(i,n){
            assert(v[i]>=0&&v[i]<128);
            nxt[v[i]].pb(i);
        }
    }
    inline int lcp(const string &v){
        int nw=0,ret=0;
        for(char i:v){
            assert(i>=0&&i<128);
            auto it=lower_bound(ALL(nxt[i]),nw);
            if(it==nxt[i].end())break;
            nw=*it+1,++ret;
        }
        return ret;
    }
    inline bool query(const string &v){
        return lcp(v)==v.size();
    }
};

template<class T,class container=unordered_map<T,int>>
struct subseq_Table{
    genID<T,container>g;
    V<V<int>>nxt;
    inline subseq_Table(const V<T> &v){
        int n=v.size();
        For(i,n){
            int k=g.get_id(v[i]);
            if(k>=nxt.size())nxt.pb(V<int>());
            nxt[k].pb(i);
        }
    }
    inline int lcp(const V<T> &v){
        int nw=0,ret=0;
        for(const T &i:v){
            if(!g.count(i))break;
            int k=g.get_id(i);
            auto it=lower_bound(ALL(nxt[k]),nw);
            if(it==nxt[k].end())break;
            nw=*it+1,++ret;
        }
        return ret;
    }
    inline bool query(const V<T> &v){
        return lcp(v)==v.size();
    }
};

struct manacher{
```

```cpp
    int n;
    V<int>p;
    inline manacher(const string &s){
        n=s.size();
        p.assign(n<<1|1,1);
        string t(n<<1|1,'#');
        For(i,n)t[i<<1|1]=s[i];
        for(int i=0,mid=-1,mx=-1;i<p.size();++i){
            if(i<=mx)p[i]=min(p[(mid<<1)-i],mx-i)+1;
            while(i>=p[i]&&i+p[i]<p.size()&&t[i-p[i]]==t[i+p[i]])++p[i];
            if(i+--p[i]>mx)mid=i,mx=i+p[i];
        }
    }
    inline int odd(int k){
        assert(0<=k&&k<n);
        return p[k<<1|1];
    }
    inline int even(int k){
        assert(0<=k&&k+1<n);
        return p[k+1<<1];
    }
    inline bool isp(int l,int r){
        assert(0<=l),assert(l<=r),assert(r<n);
        return p[l+r+1]>=r-l+1;
    }
};

inline V<int> get_kmp(const string &s){
    int n=s.size();
    V<int>kmp(n);
    for(int i=1,j=0;i<n;++i){
        while(j&&s[j]!=s[i])j=kmp[j-1];
        if(s[j]==s[i])++j;
        kmp[i]=j;
    }
    return kmp;
}
inline V<int> find_kmp(const V<int> &kmp,const string &s,const string &t){
    int n=s.size(),m=t.size();
    V<int>ret;
    for(int i=0,j=0;i<n;++i){
        while(j&&t[j]!=s[i])j=kmp[j-1];
        if(t[j]==s[i])++j;
        if(j==m)ret.pb(i);
    }
    return ret;
}
```

并查集.hpp

```cpp
struct dsu{
    V<int>fa;
    inline void resize(int n){V<int>(n,-1).swap(fa);}
    inline dsu(int n=0){resize(n);}
```

```cpp
    int find(int k){return fa[k]<0?k:fa[k]=find(fa[k]);}
    inline bool merge(int x,int y){
        x=find(x),y=find(y);
        if(x!=y)fa[x]+=fa[y],fa[y]=x;
        return x!=y;
    }
    inline bool same(int x,int y){return find(x)==find(y);}
    inline int size(int k){return -fa[find(k)];}
};
inline pair<V<V<int>>,V<int>> kruskal_tree(int n,V<array<int,3>> &e){
    int cnt=n;
    dsu d(n+n-1);
    V<V<int>>to(n+n-1);
    V<int>val(n+n-1);
    sort(ALL(e),[&](const auto &x,const auto &y){return x[2]<y[2];});
    for(const auto &i:e){
        int fx=d.find(i[0]),fy=d.find(i[1]);
        if(fx!=fy){
            d.fa[fx]=d.fa[fy]=cnt;
            to[cnt].pb(fx),to[cnt].pb(fy);
            val[cnt++]=i[2];
        }
    }
    assert(cnt==n+n-1);
    return {to,val};
}

struct range_dsu{
    V<V<int>>fa;
    int lg,n;
    inline void resize(int _n){
        V<V<int>>(lg=((n=_n)?__lg(n):-1)+1).swap(fa);
        For(i,lg)fa[i].resize(n-(1<<i)+1,-1);
    }
    inline range_dsu(int _n=0){resize(_n);}
    int find(int d,int k){return fa[d][k]<0?k:fa[d][k]=find(d,fa[d][k]);}
    inline void merge(int d,int x,int y){
        x=find(d,x),y=find(d,y);
        if(x>y)swap(x,y);
        if(x!=y)fa[d][x]+=fa[d][y],fa[d][y]=x;
    }
    inline void merge(int x1,int x2,int y1,int y2){
        assert(x2-x1==y2-y1);
        Rep(i,lg)if(x1+(1<<i)-1<=x2){
            merge(i,x1,y1);
            x1+=1<<i,y1+=1<<i;
        }
    }
    inline void init(){
        REP(i,1,lg)For(j,n-(1<<i)+1){
            int k=find(i,j);
            merge(i-1,j,k),merge(i-1,j+(1<<i-1),k+(1<<i-1));
        }
    }
}
```

```cpp
    int find(int k){return fa[0][k]<0?k:fa[0][k]=find(fa[0][k]);}
    inline bool same(int x,int y){return find(x)==find(y);}
    inline int size(int k){return -fa[0][find(k)];}
};
```

数论.hpp

```cpp
// assumed that [mod<=INT_MAX] is true

template<class T>
T exgcd(const T &a,const T &b,T &x,T &y){
    if(!b){x=1,y=0;return a;}
    T g=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return g;
};
template<class T>
inline T inv_exgcd(T n,T p=mod){
    // n*inv = 1 (mod p)
    // n*inv + p*k = 1
    // a*x + b*y = 1
    T inv=0,tmp=0;
    exgcd(n,p,inv,tmp);
    return inv<0?inv+p:inv;
}
template<class T>
inline ll exCRT(const V<T> &a,const V<T> &m){
    int n=a.size();
    assert(n==m.size());
    For(i,n)assert(0<=a[i]&&0<m[i]);
    function<ll(ll,ll,ll)>mul=[&](ll x,ll y,ll p=mod){
        ll z=0;
        auto add=[&](ll x,ll y){return x+y>=p?x+y-p:x+y;};
        for(x%=p;y;x=add(x,x),y>>=1)(y&1)&&(z=add(z,x));
        return z;
    };
    ll md=m[0],ret=a[0],x,y;
    FOR(i,1,n){
        ll g=exgcd(md,(ll)m[i],x,y),res=a[i]-ret%m[i];
        if(res<0)res+=m[i];
        if(res%g)return -1;
        ll mg=m[i]/g;
        if(x<0)x+=m[i];
        ret+=(x=mul(x,res/g,mg))*md;
        ret%=(md*=mg);
        if(ret<0)ret+=md;
    }
    return ret;
}

inline V<int> inverse(int n,int p=mod){
    V<int>inv(n+1);
    inv[1]=1;
    FOR(i,2,n+1)inv[i]=1ll*(p-p/i)*inv[p%i]%p;
```

```cpp
        return inv;
}

inline V<V<int>> comb(int n,int m=-1,int p=mod){
    if(m==-1)m=n;
    if(n<m||m<0)return V<V<int>>();
    V<V<int>>C(n+1,V<int>(m+1));
    For(i,n+1){
        C[i][0]=1;
        FOR(j,1,min(i+1,m+1)){
            C[i][j]=C[i-1][j-1]+C[i-1][j];
            if(C[i][j]>=p)C[i][j]-=p;
        }
    }
    return C;
}

struct comb_table{
    int n;
    V<mi>fac,ifac;
    inline comb_table(int n_=0){n=n_,init();}
    inline void init(){
        V<mi>(n+1).swap(fac),V<mi>(n+1).swap(ifac);
        fac[0]=1;
        FOR(i,1,n+1)fac[i]=fac[i-1]*i;
        ifac[n]=1/fac[n];
        Rep(i,n)ifac[i]=ifac[i+1]*(i+1);
    }
    inline mi C(int x,int y){return x<y||y<0?0:fac[x]*ifac[y]*ifac[x-y];}
};

struct pri_table{
    int n;
    // fac[i] is the minimum prime factor of i
    V<int>fac,pri;
    inline pri_table(int n_=0){n=n_,init();}
    inline void init(){
        if(n<1)return;
        V<int>(n+1).swap(fac),V<int>().swap(pri);
        fac[1]=1;
        FOR(i,2,n+1){
            if(!fac[i])fac[i]=i,pri.pb(i);
            for(int j:pri){
                if(i*j>n)break;
                fac[i*j]=j;
                if(i%j==0)break;
            }
        }
    }
    inline bool isp(int k){return k<2?false:(fac[k]==k);}
    inline V<int> div(int k){
        assert(k<=n);
        if(k<2)return V<int>();
        V<int>ret;
```

```cpp
        while(k>1){
            int f=fac[k];
            do k/=f;while(k%f==0);
            ret.pb(f);
        }
        return ret;
    }
};

struct mu_table{
    int n;
    V<int>mu,pri;
    V<bool>vis;
    inline mu_table(int n_=0){n=n_,init();}
    inline void init(){
        if(n<1)return;
        V<int>(n+1).swap(mu),V<int>().swap(pri),V<bool>(n+1).swap(vis);
        mu[1]=1;
        FOR(i,2,n+1){
            if(!vis[i])mu[i]=-1,pri.pb(i);
            for(int j:pri){
                if(i*j>n)break;
                vis[i*j]=true;
                if(i%j==0)break;
                mu[i*j]=-mu[i];
            }
        }
    }
    inline int get(int k){return k<1?0:mu[k];}
};

struct phi_table{
    int n;
    V<int>phi,pri;
    inline phi_table(int n_=0){n=n_,init();}
    inline void init(){
        if(n<1)return;
        V<int>(n+1).swap(phi),V<int>().swap(pri);
        phi[1]=1;
        FOR(i,2,n+1){
            if(!phi[i])phi[i]=i-1,pri.pb(i);
            for(int j:pri){
                if(i*j>n)break;
                if(i%j==0){
                    phi[i*j]=phi[i]*j;
                    break;
                }
                phi[i*j]=phi[i]*(j-1);
            }
        }
    }
    inline int get(int k){return k<1?0:phi[k];}
};
```

```cpp
struct d_table{
    int n;
    V<int>cnt,d,pri;
    V<bool>vis;
    inline d_table(int n_=0){n=n_,init();}
    inline void init(){
        if(n<1)return;
        V<int>(n+1).swap(cnt),V<int>(n+1).swap(d),V<int>().swap(pri),V<bool>(n+1↩
↪ ).swap(vis);
        cnt[1]=d[1]=1;
        FOR(i,2,n+1){
            if(!vis[i])cnt[i]=1,d[i]=2,pri.pb(i);
            for(int j:pri){
                if(i*j>n)break;
                vis[i*j]=true;
                if(i%j==0){
                    int &x=cnt[i*j];
                    x=cnt[i]+1;
                    d[i*j]=d[i]/x*(x+1);
                    break;
                }
                cnt[i*j]=1,d[i*j]=d[i]<<1;
            }
        }
    }
    inline int get(int k){return k<1?0:d[k];}
};

inline mi lagrange(int l,const V<mi> &y,int x){
    assert(y.size());
    int n=y.size();
    if(n==1)return y[0];
    if(l<=x&&x<l+n)return y[x-l];
    int r=l+n-1;
    r%=mod;if(r<0)r+=mod;
    x%=mod;if(x<0)x+=mod;
    if(r>=x&&x>r-n)return y[n-(r-x)-1];
    V<mi>ifac(n);
    ifac[0]=ifac[1]=1;
    FOR(i,2,n)ifac[i]=(mod-mod/i)*ifac[mod%i];
    FOR(i,2,n)ifac[i]*=ifac[i-1];
    V<mi>suf(n);
    suf[n-1]=1;
    REP(i,1,n)suf[i-1]=suf[i]*(x+mod-r+n-1-i);
    mi pre=1,ret=0;
    For(i,n){
        if((n-i)&1)ret+=y[i]*pre*suf[i]*ifac[i]*ifac[n-1-i];
        else ret-=y[i]*pre*suf[i]*ifac[i]*ifac[n-1-i];
        pre*=x+mod-r+n-1-i;
    }
    return ret;
}
inline mi sumexp(int n,int k){
    assert(min(n,k)>=0);
```

```cpp
    V<int>pri;
    V<mi>pw(k+2);
    pw[0]=!k,pw[1]=1;
    FOR(i,2,k+2){
        if(!pw[i])pri.pb(i),pw[i]=mi(i)^k;
        for(int j:pri){
            if(i*j>k+1)break;
            pw[i*j]=pw[i]*pw[j];
            if(i%j==0)break;
        }
    }
    FOR(i,2-!k,k+2)pw[i]+=pw[i-1];
    return lagrange(0,pw,n);
}

/*
pre_f=sum(mu) pre_g=n pre_fg=1
pre_f=sum(phi) pre_g=n pre_fg=n*(n+1)/2
pre_f=sum(phi*id) pre_g=n*(n+1)/2 pre_fg=n*(n+1)*(2n+1)/6
*/
template<class T,class container>
T du_sieve(T n,const V<T> &pre_f,const function<T(T)> &pre_g,const function<T(T)>
 ↪  &pre_fg,container &h){
    if(n<pre_f.size())return pre_f[n];
    auto it=h.emplace(n,0);
    T &x=it.fi->se;
    if(it.se){
        T pre=pre_g(1);
        x=pre_fg(n);
        for(T i=2;i<=n;++i){
            T div=n/i,j=n/div,cur=pre_g(j);
            x-=(cur-pre)*du_sieve(div,pre_f,pre_g,pre_fg,h);
            i=j,pre=cur;
        }
    }
    return x;
}

struct vote_1{
    pii v;
    inline vote_1(){v={-1,0};}
    inline vote_1(int id,int cnt=1){v={id,cnt};}
    inline vote_1 operator+(const vote_1 &rhs){
        vote_1 ret=*this;
        if(!~ret.v.fi)ret=rhs;
        else if(~rhs.v.fi){
            if(ret.v.fi==rhs.v.fi)ret.v.se+=rhs.v.se;
            else{
                if(ret.v.se<rhs.v.se)ret={rhs.v.fi,rhs.v.se-ret.v.se};
                else ret.v.se-=rhs.v.se;
            }
        }
        return ret;
    }
```

```cpp
};

template<int(*n)()>
struct vote{
    V<pii>v;
    inline vote(){V<pii>(n(),{-1,0}).swap(v);}
    inline vote(int id,int cnt=1){V<pii>(n(),{-1,0}).swap(v),v[0]={id,cnt};}
    inline vote operator+(const vote<n> &rhs){
        vote<n>ret=*this;
        for(pii i:rhs.v){
            if(!~i.fi)break;
            for(pii &j:ret.v)if(!~j.fi||i.fi==j.fi){
                j.fi=i.fi,j.se+=i.se;
                goto skip;
            }
            for(pii &j:ret.v)if(i.se>j.se)swap(i,j);
            for(pii &j:ret.v)j.se-=i.se;
            skip:;
        }
        return ret;
    }
};

template<int w2>
struct fp2{
    mi a,b;
    inline fp2(mi _a=0,mi _b=0):a(_a),b(_b){}
    inline fp2 operator+(mi rhs)const{return fp2(a+rhs,b);}
    inline fp2 operator-(mi rhs)const{return fp2(a-rhs,b);}
    inline fp2 operator*(mi rhs)const{return fp2(a*rhs,b*rhs);}
    inline fp2 operator/(mi rhs)const{mi inv=1/rhs;return fp2(a*inv,b*inv);}
    inline fp2 operator^(int k)const{fp2
    ↪   pw=*this,ret(1);for(;k;k>>=1,pw=pw*pw)if(k&1)ret=ret*pw;return ret;}
    inline fp2& operator+=(mi rhs){a+=rhs;return *this;}
    inline fp2& operator-=(mi rhs){a-=rhs;return *this;}
    inline fp2& operator*=(mi rhs){a*=rhs,b*=rhs;return *this;}
    inline fp2& operator/=(mi rhs){mi inv=1/rhs;a*=inv,b*=inv;return *this;}
    inline fp2& operator^=(int k){fp2
    ↪   tmp(1),base=*this;for(;k;k>>=1,base*=base)if(k&1)tmp*=base;return
    ↪   *this=tmp;}
    inline fp2 operator+(const fp2&rhs)const{return fp2(a+rhs.a,b+rhs.b);}
    inline fp2 operator-(const fp2&rhs)const{return fp2(a-rhs.a,b-rhs.b);}
    inline fp2 operator*(const fp2&rhs)const{return
    ↪   fp2(a*rhs.a+b*rhs.b*w2,a*rhs.b+rhs.a*b);}
    inline fp2 operator/(const fp2&rhs)const{assert(rhs.a.val||rhs.b.val);mi
    ↪   inv=1/(rhs.a*rhs.a-rhs.b*rhs.b*w2);return
    ↪   fp2((a*rhs.a-b*rhs.b*w2)*inv,(rhs.a*b-a*rhs.b)*inv);}
    inline fp2& operator+=(const fp2&rhs){a+=rhs.a,b+=rhs.b;return *this;}
    inline fp2& operator-=(const fp2&rhs){a-=rhs.a,b-=rhs.b;return *this;}
    inline fp2& operator*=(const fp2&rhs){mi
    ↪   x=a*rhs.a+b*rhs.b*w2,y=a*rhs.b+rhs.a*b;a=x,b=y;return *this;}
    inline fp2& operator/=(const fp2&rhs){assert(rhs.a.val||rhs.b.val);mi
    ↪   inv=1/(rhs.a*rhs.a-rhs.b*rhs.b*w2);mi
    ↪   x=(a*rhs.a-b*rhs.b*w2)*inv,y=(rhs.a*b-a*rhs.b)*inv;a=x,b=y;return *this;}
```

```
        inline fp2 operator-()const{return fp2(-a,-b);}
        friend fp2 operator+(mi lhs,const fp2&rhs){return fp2(lhs+rhs.a,rhs.b);}
        friend fp2 operator-(mi lhs,const fp2&rhs){return fp2(lhs-rhs.a,-rhs.b);}
        friend fp2 operator*(mi lhs,const fp2&rhs){return fp2(lhs*rhs.a,lhs*rhs.b);}
        friend fp2 operator/(mi lhs,const fp2&rhs){assert(rhs.a.val||rhs.b.val);mi
        ↪  inv=1/(rhs.a*rhs.a-rhs.b*rhs.b*w2);return
        ↪  fp2(lhs*rhs.a*inv,-lhs*rhs.b*inv);}
};
```

树.hpp

```
template<class T>
inline V<pii> cart_seq(const V<T> &v,function<bool(T,T)>cmp=[](T x,T y){return
↪  x>y;}){
    int n=v.size();
    V<pii>ret(n,pii(-1,n));
    stack<int>st;
    For(i,n){
        while(st.size()&&cmp(v[i],v[st.top()]))ret[st.top()].se=i-1,st.pop();
        if(st.size())ret[i].fi=st.top()+1;
        st.push(i);
    }
    return ret;
}
template<class T>
inline V<pii> cart_son(const V<T> &v,function<bool(T,T)>cmp=[](T x,T y){return
↪  x>y;}){
    int n=v.size();
    V<pii>ret(n,pii(-1,n));
    stack<int>st;
    For(i,n){
        while(st.size()&&cmp(v[i],v[st.top()]))ret[i].fi=st.top(),st.pop();
        if(st.size())ret[st.top()].se=i;
        st.push(i);
    }
    return ret;
}

struct lca_table{
    int n,rt;
    V<V<int>>to;
    inline void resize(int n_){V<V<int>>(n=n_).swap(to);}
    inline lca_table(int n_=0){resize(n_);}
    inline void add_edge(int x,int y){
        assert(0<=x),assert(x<n),assert(0<=y),assert(y<n),assert(x!=y);
        to[x].pb(y),to[y].pb(x);
    }
    inline lca_table(const V<V<int>>&to_){n=(to=to_).size();init();}
    V<int>dep,fa,siz,son,top;
    inline void init(int _rt=0){
        rt=_rt;
        V<int>(n).swap(dep),V<int>(n).swap(fa),V<int>(n).swap(siz),V<int>(n,-1).⌋
↪  swap(son);
        function<void(int,int)>dfs1=[&](int p,int f){
```

```
                    if(~f)dep[p]=dep[f]+1;
                    fa[p]=f,siz[p]=1;
                    for(int i:to[p])
                        if(i!=f){
                            dfs1(i,p);
                            siz[p]+=siz[i];
                            if(!~son[p]||siz[i]>siz[son[p]])son[p]=i;
                        }
                };
                V<int>(n,-1).swap(top);
                dfs1(rt,-1);
                function<void(int,int)>dfs2=[&](int p,int k){
                    top[p]=k;
                    if(~son[p]){
                        dfs2(son[p],k);
                        for(int i:to[p])
                            if(!~top[i])
                                dfs2(i,i);
                    }
                };
                dfs2(rt,rt);
        }
        inline int lca(int x,int y){
            assert(0<=x),assert(x<n),assert(0<=y),assert(y<n);
            while(top[x]!=top[y]){
                if(dep[top[x]]<dep[top[y]])swap(x,y);
                x=fa[top[x]];
            }
            return dep[x]<dep[y]?x:y;
        }
};

struct tree_chain{
    int n,rt;
    V<V<int>>to;
    inline void resize(int n_){V<V<int>>(n=n_).swap(to);}
    inline tree_chain(int n_=0){resize(n_);}
    inline void add_edge(int x,int y){
        assert(0<=x),assert(x<n),assert(0<=y),assert(y<n),assert(x!=y);
        to[x].pb(y),to[y].pb(x);
    }
    inline tree_chain(const V<V<int>>&to_){n=(to=to_).size();init();}
    V<int>dep,fa,rev,seg,siz,son,top;
    inline void init(int _rt=0){
        rt=_rt;
        V<int>(n).swap(dep),V<int>(n).swap(fa),V<int>(n).swap(siz),V<int>(n,-1).
    swap(son);
        function<void(int,int)>dfs1=[&](int p,int f){
            if(~f)dep[p]=dep[f]+1;
            fa[p]=f,siz[p]=1;
            for(int i:to[p])
                if(i!=f){
                    dfs1(i,p);
                    siz[p]+=siz[i];
```

```cpp
                if(!~son[p]||siz[i]>siz[son[p]])son[p]=i;
            }
        };
        int cnt=0;
        V<int>(n).swap(rev),V<int>(n).swap(seg),V<int>(n,-1).swap(top);
        dfs1(rt,-1);
        function<void(int,int)>dfs2=[&](int p,int k){
            seg[p]=cnt,rev[cnt++]=p,top[p]=k;
            if(~son[p]){
                dfs2(son[p],k);
                for(int i:to[p])
                    if(!~top[i])
                        dfs2(i,i);
            }
        };
        dfs2(rt,rt);
    }
    inline int lca(int x,int y)const{
        assert(0<=x),assert(x<n),assert(0<=y),assert(y<n);
        while(top[x]!=top[y]){
            if(dep[top[x]]<dep[top[y]])swap(x,y);
            x=fa[top[x]];
        }
        return dep[x]<dep[y]?x:y;
    }
    inline int kthac(int p,int k){
        assert(0<=p),assert(p<n),assert(k>=0),assert(k<=dep[p]);
        while(k>dep[p]-dep[top[p]]){
            k-=dep[p]-dep[top[p]]+1;
            p=fa[top[p]];
        }
        return rev[seg[p]-k];
    }
    inline V<pii> path(int x,int y,bool dir=0){
        assert(0<=x),assert(x<n),assert(0<=y),assert(y<n);
        V<pii>ret,ter;
        bool rv=0;
        while(top[x]!=top[y]){
            if(dep[top[x]]<dep[top[y]])rv^=1,swap(x,y);
            if(dir){
                if(rv)ter.eb(seg[top[x]],seg[x]);
                else ret.eb(seg[x],seg[top[x]]);
            }
            else (rv?ter:ret).eb(seg[top[x]],seg[x]);
            x=fa[top[x]];
        }
        if(dep[x]>dep[y])rv^=1,swap(x,y);
        if(dir){
            if(rv)ter.eb(seg[y],seg[x]);
            else ret.eb(seg[x],seg[y]);
        }
        else (rv?ret:ter).eb(seg[x],seg[y]);
        reverse(ALL(ter));
        ret.insert(ret.end(),ALL(ter));
```

```cpp
        return ret;
    }
};
inline void virt_tree(V<int> &p,const tree_chain &tc,V<V<int>> &to){
    sort(ALL(p),[&](int x,int y){return tc.seg[x]<tc.seg[y];});
    p.erase(unique(ALL(p)),p.end());
    auto add_edge=[&](int x,int y){to[x].pb(y),to[y].pb(x);};
    V<int>st;
    for(int i:p){
        if(st.size()){
            int anc=tc.lca(i,st.back());
            if(anc!=st.back()){
                while(st.size()>1&&tc.seg[anc]<tc.seg[st[st.size()-2]])add_edge(st↵
                  ↪ [st.size()-2],st.back()),st.qb();
                if(st.size()==1||tc.seg[anc]>tc.seg[st[st.size()-2]])V<int>().swap↵
                  ↪ (to[anc]),add_edge(anc,st.back()),st.back()=anc;
                 else add_edge(anc,st.back()),st.qb();
            }
        }
        V<int>().swap(to[i]),st.pb(i);
    }
    while(st.size()>1)add_edge(st[st.size()-2],st.back()),st.qb();
}

// root: n-1
inline V<int> pru2fa(const V<int> &_p){
    int n=_p.size()+2;
    V<int>deg(n),p=_p;p.pb(n-1);
    for(int i:_p)++deg[i];
    V<int>fa(n-1);
    int j=0;
    For(i,n-1){
        while(deg[j])++j;
        fa[j]=p[i];
        while(i<n-1&&!--deg[p[i]]&&p[i]<j){
            if(i+1<n-1)fa[p[i]]=p[i+1];
            ++i;
        }
        ++j;
    }
    return fa;
}
inline V<V<int>> pru2tr(const V<int> &p){
    int n=p.size()+2;
    V<int>fa=pru2fa(p);
    V<V<int>>to(n);
    For(i,n-1)to[i].pb(fa[i]),to[fa[i]].pb(i);
    return to;
}
inline V<int> fa2pru(const V<int> &fa){
    int n=fa.size()+1;
    V<int>deg(n);
    for(int i:fa)++deg[i];
    int j=0;
```

```cpp
    V<int>p(n-2);
    For(i,n-2){
        while(deg[j])++j;
        p[i]=fa[j];
        while(i<n-2&&!--deg[p[i]]&&p[i]<j){
            if(i+1<n-2)p[i+1]=fa[p[i]];
            ++i;
        }
        ++j;
    }
    return p;
}
inline V<int> tr2pru(const V<V<int>> &to){
    int n=to.size();
    V<int>fa(n-1,-1);
    queue<int>q;
    q.push(n-1);
    while(q.size()){
        int p=q.front();q.pop();
        for(int i:to[p])if(i<n-1&&!~fa[i])fa[i]=p,q.push(i);
    }
    return fa2pru(fa);
}
```

树状数组.hpp

```cpp
template<class T>
struct BIT{
    // d-indexed [-d+1,n]->[1,n+d]
    V<T>c1,c2;
    int d,n;
    inline void resize(int n_,int d_=1){
        d=d_,n=n_;
        V<T>(n+d+1).swap(c1);
        V<T>(n+d+1).swap(c2);
    }
    inline BIT(int n=0,int d=1){resize(n,d);}
    inline void add(int l,int r,const T &v){
        if(l>r)return;
        l+=d,assert(0<l),assert(l<=n+d);
        for(int i=l;i<=n+d;i+=i&-i)c1[i]+=v,c2[i]+=(l-1)*v;
        r+=d,assert(0<r),assert(r<=n+d);
        for(int i=r+1;i<=n+d;i+=i&-i)c1[i]-=v,c2[i]-=r*v;
    }
    inline void add(int k,const T &v){add(k,k,v);}
    inline T query(int l,int r){
        if(l>r)return T();
        T ret=0;
        r+=d,assert(0<r),assert(r<=n+d);
        for(int i=r;i;i^=i&-i)ret+=r*c1[i]-c2[i];
        l+=d,assert(0<l),assert(l<=n+d);
        for(int i=l-1;i;i^=i&-i)ret-=(l-1)*c1[i]-c2[i];
        return ret;
    }
```

```cpp
    inline T query(int k){return query(k,k);}
};

template<class T>
struct BIT3{
    // d-indexed [-d+1,n]->[1,n+d]
    V<T>c;
    int d,n;
    inline void resize(int n_,int d_=1){
        d=d_,n=n_;
        V<T>(n+d+1).swap(c);
    }
    inline BIT3(int n=0,int d=1){resize(n,d);}
    inline void add(int k,const T &v){
        k+=d;
        assert(1<=k),assert(k<=n+d);
        for(int i=k;i<=n+d;i+=i&-i)c[i]+=v;
    }
    inline T query(int k){
        k+=d;
        assert(1<=k),assert(k<=n+d);
        T ret=0;
        for(int i=k;i>0;i^=i&-i)ret+=c[i];
        return ret;
    }
};

template<class T>
struct BIT4{
    // d-indexed [-d+1,n]->[1,n+d]
    V<T>c;
    int d,n;
    inline void resize(int n_,int d_=1){
        d=d_,n=n_;
        V<T>(n+d+1).swap(c);
    }
    inline BIT4(int n=0,int d=1){resize(n,d);}
    inline void add(int k,const T &v){
        k+=d;
        assert(1<=k),assert(k<=n+d);
        for(int i=k;i>0;i^=i&-i)c[i]+=v;
    }
    inline T query(int k){
        k+=d;
        assert(1<=k),assert(k<=n+d);
        T ret=0;
        for(int i=k;i<=n+d;i+=i&-i)ret+=c[i];
        return ret;
    }
};
template<class T>
inline ll invpair(const T &a){
    ll ret=0;
    BIT4<int>t(*max_element(ALL(a))+1);
```

```
    for(const auto &i:a)ret+=t.query(i+1),t.add(i,1);
    return ret;
}
```

矩阵.hpp

```
template<class T>
struct matrix{
    int n,m;
    V<V<T>>a;
    inline matrix(int _n=0,int _m=0,T v=T()):n(_n),m(_m){
        V<V<T>>(n,V<T>(m,v)).swap(a);
    };
    inline V<T> &operator[](int idx){return a[idx];}
    inline const V<T> &operator[](int idx)const{return a[idx];}
    inline matrix operator*(const matrix &rhs){
        assert(m==rhs.n);
        matrix ret(n,rhs.m);
        For(i,n)For(j,rhs.m)For(k,m)ret[i][j]+=a[i][k]*rhs[k][j];
        return ret;
    }
    inline matrix trans(){
        matrix ret(m,n);
        For(i,n)For(j,m)ret[j][i]=a[i][j];
        return ret;
    }
    inline bool gauss(){
        assert(n<=m);
        int nw=0;
        For(i,n){
            if(!a[nw][i])FOR(j,nw+1,n)if(a[j][i]){swap(a[nw],a[j]);break;}
            if(a[nw][i]){
                For(j,n)if(nw!=j){
                    T coef=a[j][i]/a[nw][i];
                    FOR(k,i,m)a[j][k]-=coef*a[nw][k];
                }
                ++nw;
            }
        }
        return nw==n;
    }
    inline matrix unit(){
        assert(n==m);
        matrix ret(n,n);
        For(i,n)ret[i][i]=1;
        return ret;
    }
    inline matrix pow(ull k){
        matrix base=*this,ret=unit();
        for(;k;k>>=1,base=base*base)if(k&1)ret=ret*base;
        return ret;
    }
    inline matrix mul_pow(const matrix &rhs,ull k){
        matrix base=rhs,ret=*this;
```

```
            for(;k;k>>=1,base=base*base)if(k&1)ret=ret*base;
            return ret;
        }
};

template<class T>
struct dis_matrix{
    int n,m;
    V<V<T>>a;
    inline dis_matrix(int _n=0,int _m=0,T v=T()):n(_n),m(_m){
        assert((is_same<T,int>::value)||(is_same<T,ll>::value)||(is_same<T,ull>::
        ↪ value));
        V<V<T>>(n,V<T>(m)).swap(a);
    };
    inline V<T> &operator[](int idx){return a[idx];}
    inline const V<T> &operator[](int idx)const{return a[idx];}
    inline dis_matrix operator*(const dis_matrix &rhs){
        assert(m==rhs.n);
        dis_matrix ret(n,rhs.m,is_same<T,int>::value?inf:infl);
        For(i,n)For(j,rhs.m)For(k,m)ckmin(ret[i][j],a[i][k]+rhs[k][j]);
        return ret;
    }
    inline dis_matrix pow(ull k){
        dis_matrix base=*this,ret(n,n,is_same<T,int>::value?inf:infl);
        for(;k;k>>=1,base=base*base)if(k&1)ret=ret*base;
        return ret;
    }
};
```

离散化.hpp

```
template<class T>
struct disc{
    // 0-indexed
    vector<T>d;
    inline disc(){}
    inline void insert(const T &x){d.pb(x);}
    inline void insert(const V<T> &v){d.insert(d.end(),ALL(v));}
    inline void init(){sort(ALL(d));d.erase(unique(ALL(d)),d.end());}
    inline disc(const vector<T> &v){d=move(v);init();}
    inline int query(const T &x){return lower_bound(ALL(d),x)-d.begin();}
    inline int size(){return d.size();}
};

template<class T,class container>
struct genID{
    int cnt;
    container id;
    inline genID():cnt(0){}
    inline bool count(const T &ele){return id.count(ele);}
    inline int get_id(const T &ele){
        auto it=id.emplace(ele,-1);
        if(it.se)it.fi->se=cnt++;
        return it.fi->se;
```

```
    }
};
```

线性基.hpp

```cpp
template<class T,int n>
struct LB{
    V<T>d;
    int cnt,failed;
    inline void clear(){cnt=failed=0,V<T>(n).swap(d);}
    inline LB(){
        assert(n>0);
        assert(n<=(is_same<T,int>::value?31:is_same<T,unsigned>::value?32:is_same┘
        ↪ <T,ll>::value?63:is_same<T,ull>::value?64:-1));
        clear();
    }
    inline bool insert(T k){
        Rep(i,n)if(k>>i&1){
            if(!d[i]){
                ++cnt,d[i]=k;
                return true;
            }
            else if(!(k^=d[i]))break;
        }
        ++failed;
        return false;
    }
    inline bool can(T k){
        Rep(i,n)if(k>>i&1){
            if(!d[i])return false;
            else if(!(k^=d[i]))break;
        }
        return true;
    }
    inline T mx(T k=0){
        Rep(i,n)ckmax(k,k^d[i]);
        return k;
    }
    inline LB operator+(const LB &rhs){
        LB ret=rhs;
        ret.failed+=failed;
        For(i,n)if(d[i])ret.insert(d[i]);
        return ret;
    }
    inline LB &operator+=(const LB &rhs){
        failed+=rhs.failed;
        For(i,n)if(rhs.d[i])insert(rhs.d[i]);
        return *this;
    }
    // assumed that empty set isn't allowed
    inline T count(){return (T(1)<<cnt)-!failed;}
    // 0-indexed
    inline T rk(T k){
        T pw2=1,ret=0;
```

```
        For(i,n)if(d[i]){
            if(k>>i&1)ret|=pw2;
            pw2<<=1;
        }
        return ret;
    }
    inline T at(T k){
        if(!failed)++k;
        FOR(i,1,n)Rep(j,i)if(d[i]>>j&1)d[i]^=d[j];
        T ret=0;
        For(i,n)if(d[i]){
            if(k&1)ret^=d[i];
            k>>=1;
        }
        return k?-1:ret;
    }
};

template<class T,int n>
struct LB_ts{ // timestamp
    V<T>d;
    V<int>t;
    inline void clear(){V<T>(n).swap(d),V<int>(n).swap(t);}
    inline LB_ts(){
        assert(n>0);
        assert(n<=(is_same<T,int>::value?31:is_same<T,unsigned>::value?32:is_same↵
        ↪   <T,ll>::value?63:is_same<T,ull>::value?64:-1));
        clear();
    }
    inline bool insert(T k,int tm){
        Rep(i,n)if(k>>i&1){
            if(!d[i]){
                d[i]=k,t[i]=tm;
                return true;
            }
            else if(tm>t[i])swap(d[i],k),swap(t[i],tm);
            if(!(k^=d[i]))break;
        }
        return false;
    }
    inline bool can(T k,int tm=0){
        Rep(i,n)if(k>>i&1){
            if(!d[i]||t[i]<tm)return false;
            else if(!(k^=d[i]))break;
        }
        return true;
    }
    inline T mx(T k=0,int tm=0){
        Rep(i,n)if(t[i]>=tm)ckmax(k,k^d[i]);
        return k;
    }
    inline LB_ts operator+(const LB_ts &rhs){
        LB_ts ret=rhs;
        For(i,n)if(d[i])ret.insert(d[i],t[i]);
```

```
            return ret;
    }
    inline LB_ts &operator+=(const LB_ts &rhs){
        For(i,n)if(rhs.d[i])insert(rhs.d[i],rhs.t[i]);
        return *this;
    }
};
```

线段树.hpp

```
template<class T,T e,T(*merge)(T,T)>
struct SGT{
    int n;
    V<T>t;
    inline void resize(int n_){V<T>((n=n_)<<2,e).swap(t);}
    inline SGT(int n_=0){resize(n_);}
    inline void push_up(int p){t[p]=merge(t[p<<1],t[p<<1|1]);}
    void build(int p,int l,int r,const V<T>&v){
        if(l==r){t[p]=v[l];return;}
        int mid=l+r>>1;
        build(p<<1,l,mid,v),build(p<<1|1,mid+1,r,v);
        push_up(p);
    }
    inline void build(const V<T>&v){
        assert(v.size()==n);
        build(1,0,n-1,v);
    }
    void build(int p,int l,int r){
        if(l==r){t[p]=e;return;}
        int mid=l+r>>1;
        build(p<<1,l,mid),build(p<<1|1,mid+1,r);
        push_up(p);
    }
    inline void build(){
        build(1,0,n-1);
    }
    void query(int p,int l,int r,int ql,int qr,T &ret){
        if(ql<=l&&r<=qr){ret=merge(ret,t[p]);return;}
        int mid=l+r>>1;
        if(ql<=mid)query(p<<1,l,mid,ql,qr,ret);
        if(qr>mid)query(p<<1|1,mid+1,r,ql,qr,ret);
    }
    inline T query(int l,int r){
        assert(0<=l),assert(l<=r),assert(r<n);
        T ret=e;
        query(1,0,n-1,l,r,ret);
        return ret;
    }
    void modify(int p,int l,int r,int k,const T &v){
        if(l==r){t[p]=v;return;}
        int mid=l+r>>1;
        k<=mid?modify(p<<1,l,mid,k,v):modify(p<<1|1,mid+1,r,k,v);
        push_up(p);
    }
```

```
        inline void modify(int k,const T& v){
            assert(0<=k),assert(k<n);
            modify(1,0,n-1,k,v);
        }
    };

    template<class T,T e>
    struct SGTlazy{
        int n;
        V<T>t,tag;
        inline void resize(int n_){n=n_,V<T>(n<<2,e).swap(t),V<T>(n<<2,e).swap(tag);}
        inline SGTlazy(int n_=0){resize(n_);}
        inline void push_up(int p){}
        inline void add_tag(int p,const T &v){}
        inline void push_down(int
     ↪  p){add_tag(p<<1,tag[p]),add_tag(p<<1|1,tag[p]),tag[p]=e;}
        void build(int p,int l,int r,const V<T>&v){
            if(l==r){t[p]=v[l];return;}
            int mid=l+r>>1;
            build(p<<1,l,mid,v),build(p<<1|1,mid+1,r,v);
            push_up(p);
        }
        inline void build(const V<T>&v){
            assert(v.size()==n);
            build(1,0,n-1,v);
        }
        void query(int p,int l,int r,int ql,int qr,T &ret){
            if(ql<=l&&r<=qr){return;}
            push_down(p);
            int mid=l+r>>1;
            if(ql<=mid)query(p<<1,l,mid,ql,qr,ret);
            if(qr>mid)query(p<<1|1,mid+1,r,ql,qr,ret);
        }
        inline T query(int l,int r){
            assert(0<=l),assert(l<=r),assert(r<n);
            T ret=e;
            query(1,0,n-1,l,r,ret);
            return ret;
        }
        void modify(int p,int l,int r,int k,const T &v){
            if(l==r){t[p]=v;return;}
            push_down(p);
            int mid=l+r>>1;
            k<=mid?modify(p<<1,l,mid,k,v):modify(p<<1|1,mid+1,r,k,v);
            push_up(p);
        }
        inline void modify(int k,const T& v){
            assert(0<=k),assert(k<n);
            modify(1,0,n-1,k,v);
        }
        void add(int p,int l,int r,int ql,int qr,const T &v){
            if(ql<=l&&r<=qr){add_tag(p,v);return;}
            push_down(p);
            int mid=l+r>>1;
```

```cpp
            if(ql<=mid)add(p<<1,l,mid,ql,qr,v);
            if(qr>mid)add(p<<1|1,mid+1,r,ql,qr,v);
            push_up(p);
        }
        inline void add(int l,int r,const T &v){
            assert(0<=l),assert(l<=r),assert(r<n);
            add(1,0,n-1,l,r,v);
        }
        // int find_l(int p,int l,int r,int ql,int qr,const T &v){
        //     if(l==r)return l;
        //     push_down(p);
        //     int mid=l+r>>1;
        //     if(ql>mid||)return find_l(p<<1|1,mid+1,r,ql,qr,v);
        //     return find_l(p<<1,l,mid,ql,qr,v);
        // }
        // inline int find_l(int l,int r,const T &v){
        //     assert(0<=l),assert(l<=r),assert(r<n),assert(t[1]>=v);
        //     return find_l(1,0,n-1,l,r,v);
        // }
        // int find_r(int p,int l,int r,int ql,int qr,const T &v){
        //     if(l==r)return r;
        //     push_down(p);
        //     int mid=l+r>>1;
        //     if(qr<=mid||)return find_r(p<<1,l,mid,ql,qr,v);
        //     return find_r(p<<1|1,mid+1,r,ql,qr,v);
        // }
        // inline int find_r(int l,int r,const T &v){
        //     assert(0<=l),assert(l<=r),assert(r<n),assert(t[1]>=v);
        //     return find_r(1,0,n-1,l,r,v);
        // }
};

template<class T>
struct SGT_2n{
    int n;
    V<T>t,tag;
    inline int idx(int l,int r){return l+r|l!=r;}
    #define p idx(l,r)
    #define ls idx(l,mid)
    #define rs idx(mid+1,r)
    inline void resize(int n_){n=n_,V<T>(n<<1).swap(t),V<T>(n<<1).swap(tag);}
    inline SGT_2n(int n_=0){resize(n_);}
    void build(int l,int r,const V<T>&v){
        if(l==r){t[p]=v[l];return;}
        int mid=l+r>>1;
        build(l,mid,v),build(mid+1,r,v);
        t[p]=max(t[ls],t[rs]);
    }
    inline void build(const V<T>&v){build(0,n-1,v);}
    void modify(int l,int r,int k,const T &v){
        if(l==r){t[p]=v;return;}
        int mid=l+r>>1;
        if(tag[p])t[ls]+=tag[p],t[rs]+=tag[p],tag[ls]+=tag[p],tag[rs]+=tag[p],tag↵
          ↳  [p]=0;
```

```
            k<=mid?modify(l,mid,k,v):modify(mid+1,r,k,v);
            t[p]=max(t[ls],t[rs]);
    }
    inline void modify(int k,const T& v){modify(0,n-1,k,v);}
    #undef p
    #undef ls
    #undef rs
};
```

网络流.hpp

```
struct maxflow{
    V<pii>e;
    V<V<int>>hd;
    int n,S,T;
    inline void add_edge(int x,int y,int z){
        assert(0<=x),assert(x<n),assert(0<=y),assert(y<n),assert(z>=0);
        hd[x].pb(e.size()),e.eb(y,z),hd[y].pb(e.size()),e.eb(x,0);
    }
    inline maxflow(int _n=0,int _S=-1,int _T=-1){
        V<V<int>>(n=_n).swap(hd);
        S=_S,T=_T;
    }
    inline maxflow(const V<V<pii>> &to,int _S=-1,int _T=-1){
        V<V<int>>(n=to.size()).swap(hd);
        For(i,n)for(const pii &j:to[i])add_edge(i,j.fi,j.se);
        S=_S,T=_T;
    }
    inline ll dinic(){
        assert(S!=-1),assert(T!=-1);
        V<int>dep;
        auto bfs=[&](){
            V<int>(n).swap(dep);
            dep[S]=1;
            queue<int>q;
            q.push(S);
            while(q.size()){
                int p=q.front();q.pop();
                for(int i:hd[p])if(e[i].se&&!dep[e[i].fi])dep[e[i].fi]=dep[p]+1,q.⌋
                  ↪  push(e[i].fi);
            }
            return dep[T];
        };
        function<ll(int,ll)>dfs=[&](int p,ll lim){
            if(p==T)return lim;
            ll sum=0;
            for(int i:hd[p])if(e[i].se&&dep[p]+1==dep[e[i].fi]){
                ll f=dfs(e[i].fi,min((ll)e[i].se,lim-sum));
                e[i].se-=f,e[i^1].se+=f;
                if((sum+=f)==lim)break;
            }
            if(!sum)dep[p]=0;
            return sum;
        };
```

```cpp
        ll ret=0;
        while(bfs())ret+=dfs(S,infl);
        return ret;
    }
};

struct mincost{
    V<array<int,3>>e;
    V<V<int>>hd;
    int n,S,T;
    inline void add_edge(int x,int y,int z,int w){
        assert(0<=x),assert(x<n),assert(0<=y),assert(y<n),assert(z>=0);
        hd[x].pb(e.size()),e.pb({y,z,w}),hd[y].pb(e.size()),e.pb({x,0,-w});
    }
    inline mincost(int _n=0,int _S=-1,int _T=-1){
        V<V<int>>(n=_n).swap(hd);
        S=_S,T=_T;
    }
    inline mincost(const V<V<array<int,3>>> &to,int _S=-1,int _T=-1){
        V<V<int>>(n=to.size()).swap(hd);
        For(i,n)for(const array<int,3> &j:to[i])add_edge(i,j[0],j[1],j[2]);
        S=_S,T=_T;
    }
    typedef pair<ll,ll> pll;
    inline pll primal_dual(){
        assert(S!=-1),assert(T!=-1);
        V<ll>h;
        V<bool>vis(n);
        auto spfa=[&]{
            h.assign(n,infl);
            h[S]=0;
            queue<int>q;
            q.push(S);
            while(q.size()){
                int p=q.front();q.pop();
                vis[p]=false;
                for(int i:hd[p])if(e[i][1]&&ckmin(h[e[i][0]],h[p]+e[i][2])&&!vis[e↵
                ↪ [i][0]])q.push(e[i][0]),vis[e[i][0]]=true;
            }
        };
        spfa();
        V<ll>dis;
        V<pii>pre(n);
        auto dijkstra=[&](){
            V<ll>(n,infl).swap(dis);
            dis[S]=0;
            priority_queue<pli>q;
            q.emplace(0,S);
            V<bool>vis(n);
            while(q.size()){
                int p=q.top().se;q.pop();
                if(vis[p])continue;
                vis[p]=true;
```

```
            for(int i:hd[p])if(e[i][1]&&ckmin(dis[e[i][0]],dis[p]+e[i][2]+h[p]-
              ↪ h[e[i][0]])){
                pre[e[i][0]]={p,i};
                if(!vis[e[i][0]])q.emplace(-dis[e[i][0]],e[i][0]);
            }
        }
        return dis[T]!=infl;
    };
    ll ret1=0,ret2=0;
    while(dijkstra()){
        For(i,n)h[i]+=dis[i];
        ll f=infl;
        for(int i=T;i!=S;i=pre[i].fi)ckmin(f,(ll)e[pre[i].se][1]);
        for(int i=T;i!=S;i=pre[i].fi)e[pre[i].se][1]-=f,e[pre[i].se^1][1]+=f;
        ret1+=f,ret2+=f*h[T];
    }
    return {ret1,ret2};
}
inline pll dinic(){
    assert(S!=-1),assert(T!=-1);
    V<int>cur(n);
    V<ll>dis;
    V<V<int>>tmp=hd;
    V<bool>vis(n);
    auto spfa=[&](){
        dis.assign(n,infl);
        dis[S]=0;
        hd=tmp;
        queue<int>q;
        q.push(S);
        while(q.size()){
            int p=q.front();q.pop();
            vis[p]=false;
            for(int i:hd[p])if(e[i][1]&&ckmin(dis[e[i][0]],dis[p]+e[i][2])&&!
              ↪ vis[e[i][0]])q.push(e[i][0]),vis[e[i][0]]=true;
        }
        return dis[T]<infl;
    };
    ll ret1=0,ret2=0;
    auto dfs=[&](auto &&self,int p,ll f)->ll{
        if(p==T)return f;
        vis[p]=true;
        ll ret=0;
        while(hd[p].size()){
            int i=hd[p].back();
            if(!vis[e[i][0]]&&e[i][1]&&dis[e[i][0]]==dis[p]+e[i][2]){
                ll d=self(self,e[i][0],min((ll)e[i][1],f-ret));
                if(d){
                    ret+=d,ret2+=d*e[i][2];
                    e[i][1]-=d,e[i^1][1]+=d;
                    if(ret==f)break;
                }
            }
            hd[p].qb();
```

```
            }
            vis[p]=false;
            return ret;
        };
        while(spfa()){
            ll d;
            while(d=dfs(dfs,S,infl))ret1+=d;
        }
        return {ret1,ret2};
    }
};
```

计算几何.hpp

```cpp
const double eps=1e-8;
struct vec2D{
    double x,y;
    inline vec2D(double x_=0,double y_=0):x(x_),y(y_){}
    inline vec2D operator+(const vec2D &rhs)const{return {x+rhs.x,y+rhs.y};}
    inline vec2D operator-(const vec2D &rhs)const{return {x-rhs.x,y-rhs.y};}
    inline double cross(const vec2D &rhs){return x*rhs.y-y*rhs.x;}
    inline double operator*(const vec2D &rhs)const{return x*rhs.x+y*rhs.y;}
    // unsafe since overflow, use !cross() instead
    // inline bool coln(const vec2D &rhs){return
    ↪ (*this*rhs)*(*this*rhs)==(*this**this)*(rhs*rhs);}
    inline bool coln(const vec2D &rhs){return fabs(cross(rhs))<eps;}
    inline int dir(const vec2D &rhs){return !coln(rhs)?cross(rhs)>=eps?1:-1:0;}
    inline double norm(){return sqrt(x*x+y*y);}
    inline double proj(const vec2D &rhs){return 1.*(*this*rhs)/(*this**this);}
    inline vec2D rot(double theta){
        double c=cos(theta),s=sin(theta);
        return {x*c+y*s,y*c-x*s};
    }
    inline int quad(){
        if(x>0&&y>=0)return 1;
        if(x<=0&&y>0)return 2;
        if(x<0&&y<=0)return 3;
        if(x>=0&&y<0)return 4;
        return 0;
    }
};

struct vec2d{
    ll x,y;
    inline vec2d(ll x_=0,ll y_=0):x(x_),y(y_){}
    inline vec2d operator+(const vec2d &rhs)const{return {x+rhs.x,y+rhs.y};}
    inline vec2d operator-(const vec2d &rhs)const{return {x-rhs.x,y-rhs.y};}
    inline ll cross(const vec2d &rhs){return x*rhs.y-y*rhs.x;}
    inline ll operator*(const vec2d &rhs)const{return x*rhs.x+y*rhs.y;}
    // unsafe since overflow, use !cross() instead
    // inline bool coln(const vec2d &rhs){return
    ↪ (*this*rhs)*(*this*rhs)==(*this**this)*(rhs*rhs);}
    inline bool coln(const vec2d &rhs){return !cross(rhs);}
    inline int dir(const vec2d &rhs){return cross(rhs)?cross(rhs)>0?1:-1:0;}
```

```cpp
    inline double norm(){return sqrt(x*x+y*y);}
    inline double proj(const vec2d &rhs){return 1.*(*this*rhs)/(*this**this);}
    inline vec2D rot(double theta){
        double c=cos(theta),s=sin(theta);
        return {x*c+y*s,y*c-x*s};
    }
    inline int quad(){
        if(x>0&&y>=0)return 1;
        if(x<=0&&y>0)return 2;
        if(x<0&&y<=0)return 3;
        if(x>=0&&y<0)return 4;
        return 0;
    }
};
```