Algorithm Library

CReatiQ

South China Normal University

November 13, 2024

Contents

常用文件

DEBUG 头

```cpp
#include <bits/stdc++.h>
using namespace std;
using i64=long long;
using i128=__int128;

namespace DBG
{
    template <class T>
    void _dbg(const char *f,T t) { cerr<<f<<'='<<t<<'\n'; }

    template <class A,class... B>
    void _dbg(const char *f,A a,B... b)
    {
        while (*f!=',') cerr<<*f++;
        cerr<<'='<<a<<",";
        _dbg(f+1,b...);
    }

    template <class T>
    ostream& operator << (ostream& os,const vector<T> &v)
    {
        os<<"[ ";
        for (const auto &x:v) os<<x<<", ";
        os<<"]";
        return os;
    }

    #define dbg(...) _dbg(#__VA_ARGS__, __VA_ARGS__)
}

using namespace DBG;
```

__int128 输出流

```cpp
ostream &operator << (ostream &os,i128 n)
{
    string s;
    bool neg=n<0;
    if (neg) n=-n;
    while (n)
    {
        s+='0'+n%10;
        n/=10;
    }
    if (neg) s+='-';
    reverse(s.begin(),s.end());
    if (s.empty()) s+='0';
    return os<<s;
}
```

常用数学函数

```cpp
i64 ceilDiv(i64 n,i64 m)
{
    if (n>=0) return (n+m-1)/m;
    else return n/m;
}

i64 floorDiv(i64 n,i64 m)
{
    if (n>=0) return n/m;
    else return (n-m+1)/m;
}

i128 gcd(i128 a,i128 b)
{
```

```
    return b?gcd(b,a%b):a;
}
```

纳秒级随机种子

```
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
```

Linux 对拍

记得先 `chmod 777 check.sh`.

```
for ((i=0;i<100;i++))
do
    ./A__Generator > A.in
    ./A < A.in > A.out
    ./A__Good < A.in > A.ans

    if diff A.out A.ans;
    then
        echo "AC"
    else
        echo "WA"
        exit 1
    fi
done
```

数学

欧拉筛

时间复杂度为 $\mathcal{O}(n)$。

`phi` 为欧拉函数 $\varphi(n)$，`mu` 为莫比乌斯函数 $\mu(n)$，`d` 为约数个数 $\sigma_0(n)$，`f` 为约数和 $\sigma_1(n)$。

假如一个积性函数 f 满足：对于任意质数 p 和正整数 k，可以在 $O(1)$ 时间内计算 $f(p^k)$，那么可以在 $O(n)$ 时间内筛出 $f(1), f(2), \ldots, f(n)$ 的值。

设合数 n 的质因子分解是 $\prod_{i=1}^{k} p_i^{\alpha_i}$，其中 $p_1 < p_2 < \cdots < p_k$ 为质数，我们在线性筛中记录 $g_n = p_1^{\alpha_1}$，假如 n 被 $x \cdot p$ 筛掉（p 是质数），那么 g 满足如下递推式：

$$g_n = \begin{cases} g_x \cdot p & x \bmod p = 0 \\ p & \text{otherwise} \end{cases}$$

假如 $n = g_n$，说明 n 就是某个质数的次幂，可以 $O(1)$ 计算 $f(n)$；否则，$f(n) = f(\frac{n}{g_n}) \cdot f(g_n)$。

```
vector<int> minp,primes;
// vector<int> phi;
// vector<int> mu;
// vector<int> d,num;
// vector<int> f,g;

void sieve(int n)
{
    minp.assign(n+1,0);
    primes.clear();
    // phi.assign(n+1,0);
    // mu.assign(n+1,0);
    // d.assign(n+1,0);
    // num.assign(n+1,0);
    // f.assign(n+1,0);
    // g.assign(n+1,0);
    // phi[1]=1;
    // mu[1]=1;
    // d[1]=1;
```

```
    // f[1]=g[1]=1;
    for (int i=2;i<=n;i++)
    {
        if (!minp[i])
        {
            minp[i]=i;
            primes.push_back(i);
            // phi[i]=i-1;
            // mu[i]=-1;
            // d[i]=2;
            // num[i]=1;
            // f[i]=g[i]=i+1;
        }
        for (auto p:primes)
        {
            if (i*p>n) break;
            minp[i*p]=p;
            if (p==minp[i])
            {
                // phi[i*p]=phi[i]*p;
                // mu[i*p]=0;
                // num[i*p]=num[i]+1;
                // d[i*p]=d[i]/num[i*p]*(num[i*p]+1);
                // g[i*p]=g[i]*p+1;
                // f[i*p]=f[i]/g[i]*g[i*p];
                break;
            }
            // phi[i*p]=phi[i]*phi[p];
            // mu[i*p]=-mu[i];
            // num[i*p]=1;
            // d[i*p]=d[i]<<1;
            // f[i*p]=f[i]*f[p];
            // g[i*p]=p+1;
        }
    }
}
```

取模类（MInt）

对 MInt<0> 修改 Mod 可以起到动态模数的效果，但常数较大。

```
template <class T>
constexpr T power(T a,i64 b)
{
    T res=1;
    for (;b;b>>=1,a*=a)
        if (b&1) res*=a;
    return res;
}

template <int P>
struct MInt
{
    int x;
    constexpr MInt():x{} {}
    constexpr MInt(i64 x):x{norm(x%getMod())} {}

    static int Mod;
    constexpr static int getMod()
    {
        if (P>0) return P;
        else return Mod;
    }

    constexpr static void setMod(int Mod_) { Mod=Mod_; }

    constexpr int norm(int x) const
    {
        if (x<0) x+=getMod();
        if (x>=getMod()) x-=getMod();
        return x;
```

```
    }

    constexpr int val() const { return x; }

    explicit constexpr operator int () const { return x; }

    constexpr MInt operator - () const
    {
        MInt res;
        res.x=norm(getMod()-x);
        return res;
    }

    constexpr MInt inv() const
    {
        assert(x!=0);
        return power(*this,getMod()-2);
    }

    constexpr MInt &operator *= (MInt rhs) &
    {
        x=1ll*x*rhs.x%getMod();
        return *this;
    }

    constexpr MInt &operator += (MInt rhs) &
    {
        x=norm(x+rhs.x);
        return *this;
    }

    constexpr MInt &operator -= (MInt rhs) &
    {
        x=norm(x-rhs.x);
        return *this;
    }

    constexpr MInt &operator /= (MInt rhs) &
    {
        return *this*=rhs.inv();
    }

    friend constexpr MInt operator * (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res*=rhs;
        return res;
    }

    friend constexpr MInt operator + (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res+=rhs;
        return res;
    }

    friend constexpr MInt operator - (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res-=rhs;
        return res;
    }

    friend constexpr MInt operator / (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res/=rhs;
        return res;
    }

    friend constexpr istream &operator >> (istream &is,MInt &a)
```

```
    {
        i64 v;
        is>>v;
        a=MInt(v);
        return is;
    }

    friend constexpr ostream &operator << (ostream &os,const MInt &a) { return os<<a.val(); }

    friend constexpr bool operator == (MInt lhs,MInt rhs) { return lhs.val()==rhs.val(); }

    friend constexpr bool operator != (MInt lhs,MInt rhs) { return lhs.val()!=rhs.val(); }
};

template<>
int MInt<0>::Mod=1;

template<int V,int P>
constexpr MInt<P> CInv=MInt<P>(V).inv();
```

组合数

```
struct Comb
{
    int n;
    vector<Z> _fac,_inv,_finv;

    Comb():n{0},_fac{1},_inv{0},_finv{1}{}
    Comb(int n):Comb() { init(n); }

    void init(int m)
    {
        m=min(m,Z::getMod()-1);
        if (m<=n) return;
        _fac.resize(m+1);
        _inv.resize(m+1);
        _finv.resize(m+1);

        for (int i=n+1;i<=m;i++)
            _fac[i]=_fac[i-1]*i;
        _finv[m]=_fac[m].inv();
        for (int i=m;i>n;i--)
        {
            _finv[i-1]=_finv[i]*i;
            _inv[i]=_finv[i]*_fac[i-1];
        }
        n=m;
    }

    Z fac(int m)
    {
        if (m>n) init(m<<1);
        return _fac[m];
    }

    Z finv(int m)
    {
        if (m>n) init(m<<1);
        return _finv[m];
    }

    Z inv(int m)
    {
        if (m>n) init(m<<1);
        return _inv[m];
    }

    Z binom(int n,int m)
    {
        if (n<m||m<0) return 0;
        return fac(n)*finv(m)*finv(n-m);
```

```
    }
} comb;
```

多项式

```
vector<int> rev;
vector<Z> roots{0,1};

void dft(vector<Z> &a)
{
    int n=a.size();
    if (int(rev.size())!=n)
    {
        int k=__builtin_ctz(n)-1;
        rev.resize(n);
        for (int i=0;i<n;i++)
            rev[i]=rev[i>>1]>>1|(i&1)<<k;
    }
    for (int i=0;i<n;i++)
         if (rev[i]<i)
            swap(a[i],a[rev[i]]);
    if (int(roots.size())<n)
    {
        int k=__builtin_ctz(roots.size());
        roots.resize(n);
        while ((1<<k)<n)
        {
            Z e=power(Z(3),(P-1)>>(k+1));
            for (int i=1<<(k-1);i<(1<<k);i++)
            {
                roots[i<<1]=roots[i];
                roots[i<<1|1]=roots[i]*e;
            }
            k++;
        }
    }
    for (int k=1;k<n;k<<=1)
        for (int i=0;i<n;i+=k*2)
            for (int j=0;j<k;j++)
            {
                Z u=a[i+j],v=a[i+j+k]*roots[j+k];
                a[i+j]=u+v;
                a[i+j+k]=u-v;
            }
}

void idft(vector<Z> &a)
{
    int n=a.size();
    reverse(a.begin()+1,a.end());
    dft(a);
    Z inv=(1-P)/n;
    for (int i=0;i<n;i++) a[i]*=inv;
}

struct Poly
{
    vector<Z> a;

    Poly(){}
    explicit Poly(int size,function<Z(int)>f=[](int) { return 0; }):a(size)
    { for (int i=0;i<size;i++) a[i]=f(i); }
    Poly(const vector<Z> &a):a(a){}
    Poly(const initializer_list<Z> &a):a(a){}

    int size() const { return a.size(); }

    void resize(int n) { a.resize(n); }

    Z operator [] (int idx) const
    {
```

```
        if (idx<size()) return a[idx];
        else return 0;
    }

    Z &operator [] (int idx) { return a[idx]; }

    Poly mulxk(int k) const
    {
        auto b=a;
        b.insert(b.begin(),k,0);
        return Poly(b);
    }

    Poly modxk(int k) const
    {
        k=min(k,size());
        return Poly(vector<Z>(a.begin(),a.begin()+k));
    }

    Poly divxk(int k) const
    {
        if (size()<=k) return Poly();
        return Poly(vector<Z>(a.begin()+k,a.end()));
    }

    friend Poly operator + (const Poly &a,const Poly &b)
    {
        vector<Z> res(max(a.size(),b.size()));
        for (int i=0;i<int(res.size());i++)
            res[i]=a[i]+b[i];
        return Poly(res);
    }

    friend Poly operator - (const Poly &a,const Poly &b)
    {
        vector<Z> res(max(a.size(),b.size()));
        for (int i=0;i<int(res.size());i++)
            res[i]=a[i]-b[i];
        return Poly(res);
    }

    friend Poly operator - (const Poly &a)
    {
        vector<Z> res(a.size());
        for (int i=0;i<int(res.size());i++)
            res[i]=-a[i];
        return Poly(res);
    }

    friend Poly operator * (Poly a,Poly b)
    {
        if (!a.size()||!b.size()) return Poly();
        if (a.size()<b.size()) swap(a,b);
        if (b.size()<128)
        {
            Poly c(a.size()+b.size()-1);
            for (int i=0;i<a.size();i++)
                for (int j=0;j<b.size();j++)
                    c[i+j]+=a[i]*b[j];
            return c;
        }
        int sz=1,tot=a.size()+b.size()-1;
        while (sz<tot) sz<<=1;
        a.a.resize(sz);
        b.a.resize(sz);
        dft(a.a);
        dft(b.a);
        for (int i=0;i<sz;i++)
            a.a[i]=a[i]*b[i];
        idft(a.a);
        a.resize(tot);
```

```
        return a;
    }

    friend Poly operator * (Z a,Poly b)
    {
        for (int i=0;i<int(b.size());i++) b[i]*=a;
        return b;
    }

    friend Poly operator * (Poly a,Z b)
    {
        for (int i=0;i<int(a.size());i++) a[i]*=b;
        return a;
    }

    Poly &operator += (Poly b) { return (*this)=(*this)+b; }
    Poly &operator -= (Poly b) { return (*this)=(*this)-b; }
    Poly &operator *= (Poly b) { return (*this)=(*this)*b; }
    Poly &operator *= (Z b) { return (*this)=(*this)*b; }

    Poly deriv() const
    {
        if (a.empty()) return Poly();
        vector<Z> res(size()-1);
        for (int i=0;i<size()-1;i++)
            res[i]=(i+1)*a[i+1];
        return Poly(res);
    }

    Poly integr() const
    {
        vector<Z> res(size()+1);
        for (int i=0;i<size();i++)
            res[i+1]=a[i]/(i+1);
        return Poly(res);
    }

    Poly inv(int m) const
    {
        Poly x{a[0].inv()};
        int k=1;
        while (k<m)
        {
            k<<=1;
            x=(x*(Poly{2}-modxk(k)*x)).modxk(k);
        }
        return x.modxk(m);
    }

    Poly ln(int m) const { return (deriv()*inv(m)).integr().modxk(m); }

    Poly exp(int m) const
    {
        Poly x{1};
        int k=1;
        while (k<m)
        {
            k<<=1;
            x=(x*(Poly{1}-x.ln(k)+modxk(k))).modxk(k);
        }
        return x.modxk(m);
    }

    Poly pow(int k,int m) const
    {
        int i=0;
        while (i<size()&&a[i].val()==0) i++;
        if (i==size()||1ll*i*k>=m) return Poly(vector<Z>(m));
        Z v=a[i];
        auto f=divxk(i)*v.inv();
        return (f.ln(m-i*k)*k).exp(m-i*k).mulxk(i*k)*power(v,k);
```

```
    }

    Poly sqrt(int m) const
    {
        Poly x{1};
        int k=1;
        while (k<m)
        {
            k<<=1;
            x=(x+(modxk(k)*x.inv(k)).modxk(k))*((P+1)/2);
        }
        return x.modxk(m);
    }
    Poly mulT(Poly b) const
    {
        if (b.size()==0) return Poly();
        int n=b.size();
        reverse(b.a.begin(),b.a.end());
        return ((*this)*b).divxk(n-1);
    }

    vector<Z> eval(vector<Z> x) const
    {
        if (size()==0) return vector<Z>(x.size(),0);
        const int n=max(int(x.size()),size());
        vector<Poly> q(n<<2);
        vector<Z> ans(x.size());
        x.resize(n);
        function<void(int,int,int)> build=[&](int p,int l,int r)
        {
            if (r-l==1) q[p]=Poly{1,-x[l]};
            else
            {
                int m=(l+r)>>1;
                build(p<<1,l,m);
                build(p<<1|1,m,r);
                q[p]=q[p<<1]*q[p<<1|1];
            }
        };
        function<void(int,int,int,const Poly&)> work=[&](int p,int l,int r,const Poly &num)
        {
            if (r-l==1)
            {
                if (l<int(ans.size())) ans[l]=num[0];
            }
            else
            {
                int m=(l+r)>>1;
                work(p<<1,l,m,num.mulT(q[p<<1|1]).modxk(m-l));
                work(p<<1|1,m,r,num.mulT(q[p<<1]).modxk(r-m));
            }
        };
        build(1,0,n);
        work(1,0,n,mulT(q[1].inv(n)));
        return ans;
    }
};
```

原根表

prime	r	k	g
3	1	1	2
5	1	2	2
17	1	4	3
97	3	5	5
193	3	6	5
257	1	8	3
7681	15	9	17
12289	3	12	11
40961	5	13	3
65537	1	16	3

```
786433              3   18  10
5767169             11  19  3
7340033             7   20  3
23068673            11  21  3
104857601           25  22  3
167772161           5   25  3
469762049           7   26  3
1004535809          479 21  3
2013265921          15  27  31
2281701377          17  27  3
3221225473          3   30  5
75161927681         35  31  3
77309411329         9   33  7
206158430209        3   36  22
2061584302081       15  37  7
2748779069441       5   39  3
6597069766657       3   41  5
39582418599937      9   42  5
79164837199873      9   43  5
263882790666241     15  44  7
1231453023109121    35  45  3
1337006139375617    19  46  3
3799912185593857    27  47  5
4222124650659841    15  48  19
7881299347898369    7   50  6
31525197391593473   7   52  3
180143985094819841  5   55  6
1945555039024054273 27  56  5
4179340454199820289 29  57  3
```

线性基

```cpp
struct LB
{
    static constexpr int L=60;
    array<i64,L+1> a{};

    LB(){}

    LB(const vector<i64> &v) { init(v); }

    bool insert(i64 t)
    {
        for (int i=L;i>=0;i--)
            if (t&(1ll<<i))
            {
                if (!a[i])
                {
                    a[i]=t;
                    return 1;
                }
                else t^=a[i];
            }
        return 0;
    }

    void init(const vector<i64> &v) { for (auto x:v) insert(x); }

    bool check(i64 t)
    {
        for (int i=L;i>=0;i--)
            if (t&(1ll<<i))
                if (!a[i]) return 0;
                else t^=a[i];
        return 1;
    }

    i64 QueryMax()
    {
        i64 res=0;
        for (int i=L;i>=0;i--)
```

```
            res=max(res,res^a[i]);
        return res;
    }

    i64 QueryMin()
    {
        for (int i=0;i<=L;i++)
            if (a[i]) return a[i];
        return 0;
    }

    i64 QueryKth(int k)
    {
        i64 res=0;
        int cnt=0;
        array<i64,L+1> tmp{};
        for (int i=0;i<=L;i++)
        {
            for (int j=i-1;j>=0;j--)
                if (a[i]&(1ll<<j)) a[i]^=a[j];
            if (a[i]) tmp[cnt++]=a[i];
        }
        if (k>=(1ll<<cnt)) return -1;
        for (int i=0;i<cnt;i++)
            if (k&(1ll<<i)) res^=tmp[i];
        return res;
    }
};
```

min-plus 卷积

$\mathcal{O}(n\log n)$，但要求 b 是凸的。

```
template <class T>
vector<T> min_plus_convolution(const vector<T> &a,const vector<T> &b)
{
    int n=a.size(),m=b.size();
    vector<T> c(n+m-1);

    function<void(int,int,int,int)> solve=[&](int l,int r,int ql,int qr)
    {
        if (l>r) return;
        int mid=(l+r)>>1;
        while (ql+m<=l) ++ql;
        while (qr>r) --qr;
        int qmid=-1;
        c[mid]=inf;
        for (int i=ql;i<=qr;i++)
        {
            if (a[i]+b[mid-i]-i<c[mid])
            {
                c[mid]=a[i]+b[mid-i];
                qmid=i;
            }
            else if (mid-i>=0&&mid-i<m) qmid=i;
        }
        solve(l,mid-1,ql,mid);
        solve(mid+1,r,qmid,qr);
    };

    solve(0,n+m-2,0,n-1);
    return c;
}
```

模意义分数还原

分别是求：分子不大于 A 时分子最大的分数；分子分母最大值最小的分数。

```
pair<int,int> restore(int q,int A)
{
```

```
    int x=q,y=P,a=1,b=0;
    while (x>A)
    {
        swap(x,y);
        swap(a,b);
        a-=x/y*b;
        x%=y;
    }
    return make_pair(x,a);
}

pair<int,int> restore(int x)
{
    vector<int> a;
    int p=P;
    Z inv=Z(x).inv();
    while (x)
    {
        a.push_back(x);
        swap(x,p);
        x%=p;
    }
    pair<int,int> res{P,P};
    for (auto ca:a)
    {
        int cb=(Z(ca)*inv).x;
        ca=min(ca,P-ca);
        cb=min(cb,P-cb);
        if (max(res.first,res.second)>max(ca,cb))
            res={ca,cb};
    }
    return res;
}
```

Exgcd

可以证明 $|x| \le b, |y| \le a$。

```
void exgcd(i64 a,i64 b,i64 &x,i64 &y)
{
    if (!b)
    {
        x=1; y=0;
        return;
    }
    exgcd(b,a%b,x,y);
    swap(x,y);
    y-=a/b*x;
    return;
}
```

二元一次不定方程

给定不定方程 $ax + by = c$。

若该方程无整数解，输出 -1。

若该方程有整数解，且有正整数解，则输出其**正整数**解的数量，所有**正整数**解中 x 的最小值，所有**正整数**解中 y 的最小值，所有**正整数**解中 x 的最大值，以及所有**正整数**解中 y 的最大值。

若方程有整数解，但没有正整数解，输出所有**整数解**中 x 的最小正整数值，y 的最小正整数值。

```
void exgcd(i64 a,i64 b,i64 &x,i64 &y)
{
    if (!b)
    {
        x=1; y=0;
        return;
    }
    exgcd(b,a%b,x,y);
```

```
    swap(x,y);
    y-=a/b*x;
    return;
}

i64 ceilDiv(i64 n,i64 m)
{
    if (n>=0) return (n+m-1)/m;
    else return n/m;
}

i64 floorDiv(i64 n,i64 m)
{
    if (n>=0) return n/m;
    else return (n-m+1)/m;
}

void R()
{
    i64 a,b,c,x,y,t;
    cin>>a>>b>>c;
    t=__gcd(a,b);
    if (c%t)
    {
        cout<<"-1\n";
        return;
    }
    exgcd(a,b,x,y);
    x*=c/t,y*=c/t;
    i64 l=ceilDiv(1ll-x,b/t),r=floorDiv(y-1ll,a/t);
    if (l>r) cout<<x+l*b/t<<' '<<y-r*a/t<<'\n';
    else cout<<r-l+1ll<<' '<<x+l*b/t<<' '<<y-r*a/t<<' '<<x+r*b/t<<' '<<y-l*a/t<<'\n';
    return;
}
```

行列式求值

时间复杂度为 $\mathcal{O}(n^3)$。

```
Z det(vector<vector<Z>> a)
{
    int n=a.size(),fl=1;
    Z res=1;
    for (int i=0;i<n;i++)
    {
        for (int j=i+1;j<n;j++)
        {
            while (a[i][i].x)
            {
                int d=a[j][i].x/a[i][i].x;
                for (int k=i;k<n;k++)
                    a[j][k]-=a[i][k]*d;
                swap(a[i],a[j]);
                fl=-fl;
            }
            swap(a[i],a[j]);
            fl=-fl;
        }
    }
    for (int i=0;i<n;i++) res*=a[i][i];
    res*=fl;
    return res;
}
```

高斯消元法

返回 -1 代表无解，其余情况返回自由元数。

```
using Real=long double;
constexpr Real eps=1e-8;
```

```cpp

int Gauss(vector<vector<Real>> a,vector<Real> &x)
{
    int n=a.size(),i=0,j=0;
    for (;i<n&&j<n;i++,j++)
    {
        int mx=i;
        for (int k=i+1;k<n;k++)
            if (abs(a[k][j])>abs(a[mx][j]))
                mx=k;
        if (mx!=i) swap(a[mx],a[i]);
        if (fabs(a[i][j])<eps)
        {
            i--;
            continue;
        }
        for (int k=i+1;k<n;k++)
            if (fabs(a[k][j])>eps)
            {
                Real t=a[k][j]/a[i][j];
                for (int l=j;l<=n;l++)
                    a[k][l]-=a[i][l]*t;
                a[k][j]=0;
            }
    }
    for (int k=i;k<n;k++)
        if (fabs(a[k][j])>eps)
            return -1;//No solution
    if (i<n) return n-i;//number of free elements
    for (int k=n-1;k>=0;k--)
    {
        for (int l=k+1;l<n;l++)
        a[k][n]-=a[k][l]*x[l];
        x[k]=a[k][n]/a[k][k];
    }
    return 0;//Only one solution
}
```

枚举二进制下有 k 个 1 的数

```cpp
for (int s=(1<<k)-1,t;s<1<<n;t=s+(s&-s),s=(s&~t)>>__lg(s&-s)+1|t)
```

数据结构

并查集（启发式合并 + 带撤销）

```cpp
struct DSU
{
    int n=0;
    vector<int> fa,siz;
    stack<int> s;

    DSU(int n) { init(n); }

    void init(int n)
    {
        fa.resize(n);
        iota(fa.begin(),fa.end(),0);
        siz.assign(n,1);
        while (!s.empty()) s.pop();
    }

    int get(int x) { return fa[x]==x?x:get(fa[x]); }

    void merge(int x,int y)
    {
        x=get(x),y=get(y);
        if (x==y) return;
        if (siz[x]<siz[y]) swap(x,y);
```

```
        s.push(y),fa[y]=x,siz[x]+=siz[y];
    }

    void undo()
    {
        if (s.empty()) return;
        int y=s.top();
        s.pop();
        siz[fa[y]]-=siz[y];
        fa[y]=y;
    }

    void back(int t=0) { while (s.size()>t) undo(); }
};
```

状压 RMQ

```
template <class T,class Cmp=less<T>>
struct RMQ
{
    const Cmp cmp=Cmp();
    static constexpr unsigned B=64;
    using u64=unsigned long long;
    int n;
    vector<vector<T>> a;
    vector<T> pre,suf,ini;
    vector<u64> stk;

    RMQ() {}
    RMQ(const vector<T> &v) { init(v); }

    void init(const vector<T> &v)
    {
        n=v.size();
        pre=suf=ini=v;
        stk.resize(n);
        if (!n) return;
        const int M=(n-1)/B+1;
        const int lg=__lg(M);
        a.assign(lg+1,vector<T>(M));
        for (int i=0;i<M;i++)
        {
            a[0][i]=v[i*B];
            for (int j=1;j<B&&i*B+j<n;j++)
                a[0][i]=min(a[0][i],v[i*B+j],cmp);
        }
        for (int i=1;i<n;i++)
            if (i%B) pre[i]=min(pre[i],pre[i-1],cmp);
        for (int i=n-2;i>=0;i--)
            if (i%B!=B-1) suf[i]=min(suf[i],suf[i+1],cmp);
        for (int j=0;j<lg;j++)
            for (int i=0;i+(2<<j)<=M;i++)
                a[j+1][i]=min(a[j][i],a[j][i+(1<<j)],cmp);
        for (int i=0;i<M;i++)
        {
            const int l=i*B;
            const int r=min(1U*n,l+B);
            u64 s=0;
            for (int j=l;j<r;j++)
            {
                while (s&&cmp(v[j],v[__lg(s)+l])) s^=1ULL<<__lg(s);
                s|=1ULL<<(j-l);
                stk[j]=s;
            }
        }
    }

    //查询区间 [l,r) 的 RMQ
    T operator()(int l,int r)
    {
        if (l/B!=(r-1)/B)
```

```
        {
            T ans=min(suf[l],pre[r-1],cmp);
            l=l/B+1,r=r/B;
            if (l<r)
            {
                int k=__lg(r-l);
                ans=min({ans,a[k][l],a[k][r-(1<<k)]},cmp);
            }
            return ans;
        }
        else
        {
            int x=B*(l/B);
            return ini[__builtin_ctzll(stk[r-1]>>(l-x))+l];
        }
    }
};
```

ST 表

```
template <class T>
struct ST
{
    int n;
    vector<vector<T>> a;

    ST() {}
    ST(const vector<T> &v) { init(v); }

    void init(const vector<T> &v)
    {
        n=v.size();
        if (!n) return;
        const int lg=__lg(n);
        a.assign(lg+1,vector<T>(n));
        a[0]=v;
        for (int j=0;j<lg;j++)
            for (int i=0;i+(2<<j)<=n;i++)
                a[j+1][i]=__gcd(a[j][i],a[j][i+(1<<j)]);
    }

    T operator()(int l,int r)
    {
        int k=__lg(r-l);
        return __gcd(a[k][l],a[k][r-(1<<k)]);
    }
};
```

树状数组

```
template <class T>
struct BIT
{
    int n;
    vector<T> a;

    BIT(int n_=0) { init(n_); }

    void init(int n_)
    {
        n=n_;
        a.assign(n,T{});
    }

    void add(int x,const T &v)
    {
        for (int i=x+1;i<=n;i+=i&-i)
            a[i-1]=a[i-1]+v;
    }

```

```
    //查询区间 [0,x)
    T sum(int x)
    {
        T ans{};
        for (int i=x;i>0;i-=i&-i)
            ans=ans+a[i-1];
        return ans;
    }

    //查询区间 [l,r)
    T rangeSum(int l,int r) { return sum(r)-sum(l); }

    int select(const T &k)
    {
        int x=0;
        T cur{};
        for (int i=1<<__lg(n);i;i>>=1)
        {
            if (x+i<=n&&cur+a[x+i-1]<=k)
            {
                x+=i;
                cur=cur+a[x-1];
            }
        }
        return x;
    }
};
```

线段树

```
template <class Info,class Tag>
struct SGT
{
    int n;
    vector<Info> info;
    vector<Tag> tag;

    SGT():n(0) {}
    SGT(int n_,Info v_=Info()) { init(n_,v_); }

    template <class T>
    SGT(vector<T> init_) { init(init_); }

    void init(int n_,Info v_=Info()) { init(vector(n_,v_)); }

    template <class T>
    void init(vector<T> init_)
    {
        n=init_.size();
        info.assign(4<<__lg(n),Info());
        tag.assign(4<<__lg(n),Tag());
        function<void(int,int,int)> build=[&](int p,int l,int r)
        {
            if (r-l==1)
            {
                info[p]=init_[l];
                return;
            }
            int m=(l+r)>>1;
            build(p<<1,l,m);
            build(p<<1|1,m,r);
            pushup(p);
        };
        build(1,0,n);
    }

    void pushup(int p) { info[p]=info[p<<1]+info[p<<1|1]; }

    void apply(int p,const Tag &v)
    {
        info[p].apply(v);
```

```
        tag[p].apply(v);
    }

    void pushdown(int p)
    {
        apply(p<<1,tag[p]);
        apply(p<<1|1,tag[p]);
        tag[p]=Tag();
    }

    void modify(int p,int l,int r,int x,const Info &v)
    {
        if (r-l==1)
        {
            info[p]=v;
            return;
        }
        int m=(l+r)>>1;
        pushdown(p);
        if (x<m) modify(p<<1,l,m,x,v);
        else modify(p<<1|1,m,r,x,v);
        pushup(p);
    }

    //O(log n) 单点修改
    void modify(int p,const Info &v) { modify(1,0,n,p,v); }

    Info rangeQuery(int p,int l,int r,int x,int y)
    {
        if (l>=y||r<=x) return Info();
        if (l>=x&&r<=y) return info[p];
        int m=(l+r)>>1;
        pushdown(p);
        return rangeQuery(p<<1,l,m,x,y)+rangeQuery(p<<1|1,m,r,x,y);
    }

    //O(log n) 区间查询 [l,r)
    Info rangeQuery(int l,int r) { rangeQuery(1,0,n,l,r); }

    void rangeApply(int p,int l,int r,int x,int y,const Tag &v)
    {
        if (l>=y||r<=x) return;
        if (l>=x&&r<=y)
        {
            apply(p,v);
            return;
        }
        int m=(l+r)>>1;
        pushdown(p);
        rangeApply(p<<1,l,m,x,y,v);
        rangeApply(p<<1|1,m,r,x,y,v);
        pushup(p);
    }

    //O(log n) 区间操作 [l,r)
    void rangeApply(int l,int r,const Tag &v) { rangeApply(1,0,n,l,r,v); }

    //O(log n) 区间 [l,r) 内查找第一个合法位置
    template <class F>
    int findFirst(int p,int l,int r,int x,int y,F pred)
    {
        if (l>=y||r<=x||!pred(info[p])) return -1;
        if (r-l==1) return l;
        int m=(l+r)>>1;
        pushdown(p);
        int res=findFirst(p<<1,l,m,x,y,pred);
        if (res==-1) res=findFirst(p<<1|1,m,r,x,y,pred);
        return res;
    }

    template <class F>
```

```
    int findFirst(int l,int r,F pred) { return findFirst(1,0,n,l,r,pred); }

    template <class F>
    int findLast(int p,int l,int r,int x,int y,F pred)
    {
        if (l>=y||r<=x||!pred(info[p])) return -1;
        if (r-l==1) return l;
        int m=(l+r)>>1;
        pushdown(p);
        int res=findFirst(p<<1|1,m,r,x,y,pred);
        if (res==-1) res=findFirst(p<<1,l,m,x,y,pred);
        return res;
    }

    template <class F>
    int findLast(int l,int r,F pred) { return findLast(1,0,n,l,r,pred); }
};

//这里默认乘法优先 (x*a+b)*c+d=x*(a*c)+(b*c+d)
struct Tag
{
    i64 a=1,b=0;
    void apply(Tag t)
    {
        a*=t.a;
        b=b*t.a+t.b;
    }
};

struct Info
{
    i64 x=0,l=0,r=0;
    void apply(Tag t)
    {
        int len=r-l+1;
        x=x*t.a+len*t.b;
    }
};

Info operator + (Info a,Info b)
{
    return {a.x+b.x,min(a.l,b.l),max(a.r,b.r)};
}
```

字符串

字符串哈希（随机模数）

```
bool isPrime(int n)
{
    if (n<=1) return 0;
    for (int i=2;i*i<=n;i++)
        if (n%i==0) return 0;
    return 1;
}

int findPrime(int n)
{
    while (!isPrime(n)) n++;
    return n;
}

mt19937 rng(time(0));
const int P=findPrime(rng()%900000000+100000000);
struct StrHash
{
    int n;
    vector<int> h,p;

    StrHash(const string &s){ init(s); }
```

```
    void init(const string &s)
    {
        n=s.size();
        h.resize(n+1);
        p.resize(n+1);
        p[0]=1;
        for (int i=0;i<n;i++) h[i+1]=(10ll*h[i]+s[i]-'a')%P;
        for (int i=0;i<n;i++) p[i+1]=10ll*p[i]%P;
    }

    //查询 [l,r) 的区间哈希
    int get(int l,int r) { return (h[r]+1ll*(P-h[l])*p[r-l])%P; }
};
```

KMP

```
vector<int> KMP(const string &s)
{
    int now=0;
    vector<int> pre(s.size(),0);
    for (int i=1;i<s.size();i++)
    {
        while (now&&s[i]!=s[now]) now=pre[now-1];
        if (s[i]==s[now]) now++;
        pre[i]=now;
    }
    return pre;
}
```

Z 函数

```
vector<int> zFunction(string s)
{
    int n=s.size();
    vector<int> z(n);
    z[0]=n;
    for (int i=1,j=1;i<n;i++)
    {
        z[i]=max(0,min(j+z[j]-i,z[i-j]));
        while (i+z[i]<n&&s[z[i]]==s[i+z[i]]) z[i]++;
        if (i+z[i]>j+z[j]) j=i;
    }
    return z;
}
```

AC 自动机

```
struct ACAM
{
    static constexpr int ALPHABET=26;
    struct Node
    {
        int len;
        int link;
        array<int,ALPHABET> next;
        Node():len{0},link{0},next{}{}
    };

    vector<Node> t;

    ACAM() { init(); }

    void init()
    {
        t.assign(2,Node());
        t[0].next.fill(1);
        t[0].len=-1;
    }
```

```
    int newNode()
    {
        t.emplace_back();
        return t.size()-1;
    }

    int add(const string &a)
    {
        int p=1;
        for (auto c:a)
        {
            int x=c-'a';
            if (t[p].next[x]==0)
            {
                t[p].next[x]=newNode();
                t[t[p].next[x]].len=t[p].len+1;
            }
            p=t[p].next[x];
        }
        return p;
    }

    void work()
    {
        queue<int> q;
        q.push(1);
        while (!q.empty())
        {
            int x=q.front();
            q.pop();
            for (int i=0;i<ALPHABET;i++)
            {
                if (t[x].next[i]==0) t[x].next[i]=t[t[x].link].next[i];
                else
                {
                    t[t[x].next[i]].link=t[t[x].link].next[i];
                    q.push(t[x].next[i]);
                }
            }
        }
    }

    int next(int p,int x) { return t[p].next[x]; }

    int link(int p) { return t[p].link; }

    int size() { return t.size(); }
};
```

后缀数组

```
struct SA
{
    int n;
    vector<int> sa,rk,lc;
    SA(const string &s)
    {
        n=s.length();
        sa.resize(n);
        rk.resize(n);
        lc.resize(n-1);
        iota(sa.begin(),sa.end(),0);
        sort(sa.begin(),sa.end(),[&](int a,int b){ return s[a]<s[b]; });
        rk[sa[0]]=0;
        for (int i=1;i<n;i++) rk[sa[i]]=rk[sa[i-1]]+(s[sa[i]]!=s[sa[i-1]]);
        int k=1;
        vector<int> tmp,cnt(n);
        tmp.reserve(n);
        while (rk[sa[n-1]]<n-1)
        {
```

```
            tmp.clear();
            for (int i=0;i<k;i++) tmp.push_back(n-k+i);
            for (auto i:sa)
                if (i>=k) tmp.push_back(i-k);
            fill(cnt.begin(),cnt.end(),0);
            for (int i=0;i<n;i++) cnt[rk[i]]++;
            for (int i=1;i<n;i++) cnt[i]+=cnt[i-1];
            for (int i=n-1;i>=0;i--) sa[--cnt[rk[tmp[i]]]]=tmp[i];
            swap(rk,tmp);
            rk[sa[0]]=0;
            for (int i=1;i<n;i++)
                rk[sa[i]]=rk[sa[i-1]]+(tmp[sa[i-1]]<tmp[sa[i]]||sa[i-1]+k==n||tmp[sa[i-1]+k]<tmp[sa[i]+k]);
            k<<=1;
        }
        for (int i=0,j=0;i<n;i++)
        {
            if (rk[i]==0) j=0;
            else
            {
                for (j-=j>0;i+j<n&&sa[rk[i]-1]+j<n&&s[i+j]==s[sa[rk[i]-1]+j];) j++;
                lc[rk[i]-1]=j;
            }//lc[i]:lcp(sa[i],sa[i+1]),lcp(sa[i],sa[j])=min{lc[i...j-1]}
        }
    }
};
```

（广义）后缀自动机

```
struct SAM
{
    static constexpr int ALPHABET=26;
    struct Node
    {
        int len;
        int link;
        array<int,ALPHABET> next;
        Node():len{},link{},next{} {}
    };

    vector<Node> t;

    SAM() { init(); }

    void init()
    {
        t.assign(2,Node());
        t[0].next.fill(1);
        t[0].len=-1;
    }

    int newNode()
    {
        t.emplace_back();
        return t.size()-1;
    }

    int extend(int lst,int c)
    {
        if (t[lst].next[c]&&t[t[lst].next[c]].len==t[lst].len+1)
            return t[lst].next[c];
        int p=lst,np=newNode(),flag=0;
        t[np].len=t[p].len+1;
        while (!t[p].next[c])
        {
            t[p].next[c]=np;
            p=t[p].link;
        }
        if (!p)
        {
            t[np].link=1;
            return np;
```

```
        }
        int q=t[p].next[c];
        if (t[q].len==t[p].len+1)
        {
            t[np].link=q;
            return np;
        }
        if (p==lst) flag=1,np=0,t.pop_back();
        int nq=newNode();
        t[nq].link=t[q].link;
        t[nq].next=t[q].next;
        t[nq].len=t[p].len+1;
        t[q].link=t[np].link=nq;
        while (p&&t[p].next[c]==q)
        {
            t[p].next[c]=nq;
            p=t[p].link;
        }
        return flag?nq:np;
    }

    int add(const string &a)
    {
        int p=1;
        for (auto c:a) p=extend(p,c-'a');
        return p;
    }

    int next(int p,int x) { return t[p].next[x]; }

    int link(int p) { return t[p].link; }

    int len(int p) { return t[p].len; }

    int size() { return t.size(); }
};
```

Manacher

```
vector<int> manacher(vector<int> s)
{
    vector<int> t{0};
    for (auto c:s)
    {
        t.push_back(c);
        t.push_back(0);
    }
    int n=t.size();
    vector<int> r(n);
    for (int i=0,j=0;i<n;i++)
    {
        if (j*2-i>=0&&j+r[j]>i) r[i]=min(r[j*2-i],j+r[j]-i);
        while (i-r[i]>=0&&i+r[i]<n&&t[i-r[i]]==t[i+r[i]]) r[i]++;
        if (i+r[i]>j+r[j]) j=i;
    }
    return r;
}
```

回文自动机

```
struct PAM
{
    static constexpr int ALPHABET_SIZE=28;
    struct Node
    {
        int len,link,cnt;
        array<int,ALPHABET_SIZE> next;
        Node():len{},link{},cnt{},next{}{}
    };
    vector<Node> t;
```

```
    int suff;
    string s;

    PAM() { init(); }

    void init()
    {
        t.assign(2,Node());
        t[0].len=-1;
        suff=1;
        s.clear();
    }

    int newNode()
    {
        t.emplace_back();
        return t.size()-1;
    }

    bool add(char c,char offset='a')
    {
        int pos=s.size();
        s+=c;
        int let=c-offset;
        int cur=suff,curlen=0;
        while (1)
        {
            curlen=t[cur].len;
            if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos]) break;
            cur=t[cur].link;
        }
        if (t[cur].next[let])
        {
            suff=t[cur].next[let];
            return 0;
        }
        int num=newNode();
        suff=num;
        t[num].len=t[cur].len+2;
        t[cur].next[let]=num;
        if (t[num].len==1)
        {
            t[num].link=t[num].cnt=1;
            return 1;
        }
        while (1)
        {
            cur=t[cur].link;
            curlen=t[cur].len;
            if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos])
            {
                t[num].link=t[cur].next[let];
                break;
            }
        }
        t[num].cnt=t[t[num].link].cnt+1;
        return 1;
    }
};
```

图论

Dijkstra

注意设定合适的 inf。

```
vector<i64> dijk(const vector<vector<pair<int,i64>>> &adj,int s)
{
    int n=adj.size();
    using pa=pair<i64,int>;
```

```
    vector<i64> d(n,inf);
    vector<int> ed(n);
    priority_queue<pa,vector<pa>,greater<pa>> q;
    q.push({0,s}); d[s]=0;
    while (!q.empty())
    {
        int u=q.top().second;
        q.pop();
        ed[u]=1;
        for (auto [v,w]:adj[u])
            if (d[u]+w<d[v])
            {
                d[v]=d[u]+w;
                q.push({d[v],v});
            }
        while (!q.empty()&&ed[q.top().second]) q.pop();
    }
    return d;
}
```

SPFA

注意设定合适的 inf。

```
vector<i64> spfa(const vector<vector<pair<int,i64>>> &adj,int s)
{
    int n=adj.size();
    assert(n);
    queue<int> q;
    vector<int> len(n),ed(n);
    vector<i64> d(n,inf);
    q.push(s); d[s]=0;
    while (!q.empty())
    {
        int u=q.front();
        q.pop();
        ed[u]=0;
        for (auto [v,w]:adj[u])
            if (d[u]+w<d[v])
            {
                d[v]=d[u]+w;
                len[v]=len[u]+1;
                if (len[v]>n) return {};
                if (!ed[v]) ed[v]=1,q.push(v);
            }
    }
    return d;
}
```

Johnson

```
vector<vector<i64>> dijk(const vector<vector<pair<int,i64>>> &adj)
{
    vector<vector<i64>> res;
    for (int i=0;i<adj.size();i++)
        res.push_back(dijk(adj,i));
    return res;
}

vector<i64> spfa(const vector<vector<pair<int,i64>>> &adj)
{
    int n=adj.size();
    assert(n);
    queue<int> q;
    vector<int> len(n),ed(n,1);
    vector<i64> d(n);
    for (int i=0;i<n;i++) q.push(i);
    while (!q.empty())
    {
        int u=q.front();
```

```
        q.pop();
        ed[u]=0;
        for (auto [v,w]:adj[u])
            if (d[u]+w<d[v])
            {
                d[v]=d[u]+w;
                len[v]=len[u]+1;
                if (len[v]>n) return {};
                if (!ed[v]) ed[v]=1,q.push(v);
            }
    }
    return d;
}

vector<vector<i64>> john(vector<vector<pair<int,i64>>> adj)
{
    int n=adj.size();
    assert(n);
    auto h=spfa(adj);
    if (!h.size()) return {};
    for (int u=0;u<n;u++)
        for (auto &[v,w]:adj[u])
            w+=h[u]-h[v];
    auto res=dijk(adj);
    for (int u=0;u<n;u++)
        for (int v=0;v<n;v++)
            if (res[u][v]!=inf)
                res[u][v]-=h[u]-h[v];
    return res;
}
```

强连通分量

```
struct SCC
{
    int n,cur,cnt;
    vector<vector<int>> adj;
    vector<int> stk,dfn,low,bel;

    SCC() {}
    SCC(int n) { init(n); }

    void init(int n)
    {
        this->n=n;
        adj.assign(n,{});
        stk.clear();
        dfn.assign(n,-1);
        low.resize(n);
        bel.assign(n,-1);
        cur=cnt=0;
    }

    void add(int u,int v) { adj[u].push_back(v); }

    void dfs(int x)
    {
        dfn[x]=low[x]=cur++;
        stk.push_back(x);
        for (auto y:adj[x])
        {
            if (dfn[y]==-1)
            {
                dfs(y);
                low[x]=min(low[x],low[y]);
            }
            else if (bel[y]==-1) low[x]=min(low[x],dfn[y]);
        }
        if (dfn[x]==low[x])
        {
            int y;
```

```
            do
            {
                y=stk.back();
                bel[y]=cnt;
                stk.pop_back();
            } while (y!=x);
            cnt++;
        }
    }

    vector<int> work()
    {
        for (int i=0;i<n;i++)
            if (dfn[i]==-1) dfs(i);
        return bel;
    }

    struct Graph
    {
        int n;
        vector<pair<int,int>> edges;
        vector<int> siz,cnte;
    };

    Graph compress()
    {
        Graph G;
        G.n=cnt;
        G.siz.resize(cnt);
        G.cnte.resize(cnt);
        for (int i=0;i<n;i++)
        {
            G.siz[bel[i]]++;
            for (auto j:adj[i])
                if (bel[i]!=bel[j])
                    G.edges.emplace_back(bel[j],bel[i]);
        }
        return G;
    };
};
```

边双连通分量

```
struct EBCC
{
    int n;
    vector<vector<int>> adj;
    vector<int> stk,dfn,low,bel;
    int cur,cnt;

    EBCC() {}
    EBCC(int n) { init(n); }

    void init(int n)
    {
        this->n=n;
        adj.assign(n,{});
        dfn.assign(n,-1);
        low.resize(n);
        bel.assign(n,-1);
        stk.clear();
        cur=cnt=0;
    }

    void add(int u,int v)
    {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void dfs(int x,int p)
```

```
    {
        dfn[x]=low[x]=cur++;
        stk.push_back(x);
        for (auto y:adj[x])
        {
            if (y==p) continue;
            if (dfn[y]==-1)
            {
                dfs(y,x);
                low[x]=min(low[x],low[y]);
            }
            else if (bel[y]==-1&&dfn[y]<dfn[x]) low[x]=min(low[x],dfn[y]);
        }
        if (dfn[x]==low[x])
        {
            int y;
            do
            {
                y=stk.back();
                bel[y]=cnt;
                stk.pop_back();
            } while (y!=x);
            cnt++;
        }
    }

    vector<int> work()
    {
        dfs(0,-1);
        return bel;
    }

    struct Graph
    {
        int n;
        vector<pair<int,int>> edges;
        vector<int> siz,cnte;
    };

    Graph compress()
    {
        Graph G;
        G.n=cnt;
        G.siz.resize(cnt);
        G.cnte.resize(cnt);
        for (int i=0;i<n;i++)
        {
            G.siz[bel[i]]++;
            for (auto j:adj[i])
            {
                if (bel[i]<bel[j]) G.edges.emplace_back(bel[i],bel[j]);
                else if (i<j) G.cnte[bel[i]]++;
            }
        }
        return G;
    };
};
```

轻重链剖分

```
struct HLD
{
    int n;
    vector<int> siz,top,dep,pa,in,out,seq;
    vector<vector<int>> adj;
    int cur;

    HLD(){}
    HLD(int n) { init(n); }

    void init(int n)
```

```
    {
        this->n=n;
        siz.resize(n);
        top.resize(n);
        dep.resize(n);
        pa.resize(n);
        in.resize(n);
        out.resize(n);
        seq.resize(n);
        cur=0;
        adj.assign(n,{});
    }

    void addEdge(int u,int v)
    {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void work(int rt=0)
    {
        top[rt]=rt;
        dep[rt]=0;
        pa[rt]=-1;
        dfs1(rt);
        dfs2(rt);
    }

    void dfs1(int u)
    {
        if (pa[u]!=-1) adj[u].erase(find(adj[u].begin(),adj[u].end(),pa[u]));
        siz[u]=1;
        for (auto &v:adj[u])
        {
            pa[v]=u;
            dep[v]=dep[u]+1;
            dfs1(v);
            siz[u]+=siz[v];
            if (siz[v]>siz[adj[u][0]])
                swap(v,adj[u][0]);
        }
    }

    void dfs2(int u)
    {
        in[u]=cur++;
        seq[in[u]]=u;
        for (auto v:adj[u])
        {
            top[v]=(v==adj[u][0])?top[u]:v;
            dfs2(v);
        }
        out[u]=cur;
    }

    int lca(int u,int v)
    {
        while (top[u]!=top[v])
        {
            if (dep[top[u]]>dep[top[v]]) u=pa[top[u]];
            else v=pa[top[v]];
        }
        return dep[u]<dep[v]?u:v;
    }

    int dist(int u,int v) { return dep[u]+dep[v]-(dep[lca(u,v)]<<1); }

    int jump(int u,int k)
    {
        if (dep[u]<k) return -1;
        int d=dep[u]-k;
```

```
        while (dep[top[u]]>d) u=pa[top[u]];
        return seq[in[u]-dep[u]+d];
    }

    bool isAncester(int u,int v) { return in[u]<=in[v]&&in[v]<out[u]; }

    int rootedParent(int u,int v)//u->root,v->point
    {
        if (u==v) return u;
        if (!isAncester(v,u)) return pa[v];
        auto it=upper_bound(adj[v].begin(),adj[v].end(),u,[&](int x,int y){ return in[x]<in[y]; })-1;
        return *it;
    }

    int rootedSize(int u,int v)//same as rootedParent
    {
        if (u==v) return n;
        if (!isAncester(v,u)) return siz[v];
        return n-siz[rootedParent(u,v)];
    }

    int rootedLca(int a,int b,int c) { return lca(a,b)^lca(b,c)^lca(c,a); }
};
```

2-SAT

```
struct TwoSat
{
    int n;
    vector<vector<int>> e;
    vector<bool> ans;

    TwoSat(int n):n(n),e(n<<1),ans(n){}

    void addClause(int u,bool f,int v,bool g)
    {
        e[u*2+!f].push_back(v*2+g);
        e[v*2+!g].push_back(u*2+f);
    }

    bool satisfiable()
    {
        vector<int> id(n*2,-1),dfn(n*2,-1),low(n*2,-1),stk;
        int now=0,cnt=0;
        function<void(int)> tarjan=[&](int u)
        {
            stk.push_back(u);
            dfn[u]=low[u]=now++;
            for (auto v:e[u])
            {
                if (dfn[v]==-1)
                {
                    tarjan(v);
                    low[u]=min(low[u],low[v]);
                }
                else if (id[v]==-1)
                    low[u]=min(low[u],dfn[v]);
            }
            if (dfn[u]==low[u])
            {
                int v;
                do
                {
                    v=stk.back();
                    stk.pop_back();
                    id[v]=cnt;
                } while (v!=u);
                cnt++;
            }
        };
        for (int i=0;i<n*2;i++)
```

```
            if (dfn[i]==-1)
                tarjan(i);
        for (int i=0;i<n;i++)
        {
            if (id[i*2]==id[i*2+1]) return 0;
            ans[i]=id[i*2]>id[i*2+1];
        }
        return 1;
    }
    vector<bool> answer() { return ans; }
};
```

最大流

```
template <class T>
struct MaxFlow
{
    struct _Edge
    {
        int to;
        T cap;
        _Edge(int to,T cap):to(to),cap(cap){}
    };

    int n;
    vector<_Edge> e;
    vector<vector<int>> g;
    vector<int> cur,h;

    MaxFlow(){}
    MaxFlow(int n) { init(n); }

    void init(int n)
    {
        this->n=n;
        e.clear();
        g.assign(n,{});
        cur.resize(n);
        h.resize(n);
    }

    bool bfs(int s,int t)
    {
        h.assign(n,-1);
        queue<int> que;
        h[s]=0;
        que.push(s);
        while (!que.empty())
        {
            const int u=que.front();
            que.pop();
            for (int i:g[u])
            {
                auto [v,c]=e[i];
                if (c>0&&h[v]==-1)
                {
                    h[v]=h[u]+1;
                    if (v==t) return 1;
                    que.push(v);
                }
            }
        }
        return 0;
    }

    T dfs(int u,int t,T f)
    {
        if (u==t) return f;
        auto r=f;
        for (int &i=cur[u];i<int(g[u].size());i++)
        {
```

```
            const int j=g[u][i];
            auto [v,c]=e[j];
            if (c>0&&h[v]==h[u]+1)
            {
                auto a=dfs(v,t,min(r,c));
                e[j].cap-=a;
                e[j^1].cap+=a;
                r-=a;
                if (r==0) return f;
            }
        }
        return f-r;
    }

    void addEdge(int u,int v,T c)
    {
        g[u].push_back(e.size());
        e.emplace_back(v,c);
        g[v].push_back(e.size());
        e.emplace_back(u,0);
    }

    T flow(int s,int t)
    {
        T ans=0;
        while (bfs(s,t))
        {
            cur.assign(n,0);
            ans+=dfs(s,t,numeric_limits<T>::max());
        }
        return ans;
    }

    vector<bool> minCut()
    {
        vector<bool> c(n);
        for (int i=0;i<n;i++) c[i]=(h[i]!=-1);
        return c;
    }

    struct Edge
    {
        int from;
        int to;
        T cap;
        T flow;
    };

    vector<Edge> edges()
    {
        vector<Edge> a;
        for (int i=0;i<e.size();i+=2)
        {
            Edge x;
            x.from=e[i+1].to;
            x.to=e[i].to;
            x.cap=e[i].cap+e[i+1].cap;
            x.flow=e[i+1].cap;
            a.push_back(x);
        }
        return a;
    }
};
```

最小费用最大流

```
template <class T>
struct MinCostFlow
{
    struct _Edge
    {
```

```
        int to;
        T cap;
        T cost;
        _Edge(int to,T cap,T cost):to(to),cap(cap),cost(cost){}

    };

    int n;
    vector<_Edge> e;
    vector<vector<int>> g;
    vector<T> h,dis;
    vector<int> pre;

    bool john(int s,int t)
    {
        dis.assign(n,numeric_limits<T>::max());
        pre.assign(n,-1);
        priority_queue<pair<T,int>,vector<pair<T,int>>,greater<pair<T,int>>> q;
        dis[s]=0;
        q.emplace(0,s);
        while (!q.empty())
        {
            T d=q.top().first;
            int u=q.top().second;
            q.pop();
            if (dis[u]!=d) continue;
            for (int i:g[u])
            {
                int v=e[i].to;
                T cap=e[i].cap;
                T cost=e[i].cost;
                if (cap>0&&dis[v]>d+h[u]-h[v]+cost)
                {
                    dis[v]=d+h[u]-h[v]+cost;
                    pre[v]=i;
                    q.emplace(dis[v],v);
                }
            }
        }
        return dis[t]!=numeric_limits<T>::max();
    }

    MinCostFlow(){}
    MinCostFlow(int n) { init(n); }

    void init(int n_)
    {
        n=n_;
        e.clear();
        g.assign(n,{});
    }

    void addEdge(int u,int v,T cap,T cost)
    {
        g[u].push_back(e.size());
        e.emplace_back(v,cap,cost);
        g[v].push_back(e.size());
        e.emplace_back(u,0,-cost);
    }

    pair<T,T> flow(int s,int t)
    {
        T flow=0;
        T cost=0;
        h.assign(n,0);
        while (john(s,t))
        {
            for (int i=0;i<n;i++) h[i]+=dis[i];
            T aug=numeric_limits<int>::max();
            for (int i=t;i!=s;i=e[pre[i]^1].to)
                aug=min(aug,e[pre[i]].cap);
```

```
            for (int i=t;i!=s;i=e[pre[i]^1].to)
            {
                e[pre[i]].cap-=aug;
                e[pre[i]^1].cap+=aug;
            }
            flow+=aug;
            cost+=aug*h[t];
        }
        return make_pair(flow,cost);
    }

    struct Edge
    {
        int from;
        int to;
        T cap;
        T cost;
        T flow;
    };

    vector<Edge> edges()
    {
        vector<Edge> a;
        for (int i=0;i<e.size();i+=2)
        {
            Edge x;
            x.from=e[i+1].to;
            x.to=e[i].to;
            x.cap=e[i].cap+e[i+1].cap;
            x.cost=e[i].cost;
            x.flow=e[i+1].cap;
            a.push_back(x);
        }
        return a;
    }
};
```

计算几何

EPS

```
const double eps=1e-8;
int sgn(double x)
{
    if (fabs(x)<eps) return 0;
    if (x>0) return 1;
    return -1;
}
```

Point

```
template <class T>
struct Point
{
    T x,y;
    Point(T x_=0,T y_=0):x(x_),y(y_) {}

    Point &operator += (Point p) &
    {
        x+=p.x;
        y+=p.y;
        return *this;
    }

    Point &operator -= (Point p) &
    {
        x-=p.x;
        y-=p.y;
        return *this;
```

```
    }

    Point &operator *= (T v) &
    {
        x*=v;
        y*=v;
        return *this;
    }

    Point operator - () const { return Point(-x,-y); }

    friend Point operator + (Point a,Point b) { return a+=b; }
    friend Point operator - (Point a,Point b) { return a-=b; }
    friend Point operator * (Point a,T b) { return a*=b; }
    friend Point operator * (T a,Point b) { return b*=a; }

    friend bool operator == (Point a,Point b) { return a.x==b.x&&a.y==b.y; }

    friend istream &operator >> (istream &is,Point &p) { return is>>p.x>>p.y; }

    friend ostream &operator << (ostream &os,Point p) { return os<<'('<<p.x<<','<<p.y<<')'; }
};

template <class T>
int sgn(const Point<T> &a) { return a.y>0||(a.y==0&&a.x>0)?1:-1; }

template <class T>
T dot(Point<T> a,Point<T> b) { return a.x*b.x+a.y*b.y; }

template <class T>
T cross(Point<T> a,Point<T> b) { return a.x*b.y-a.y*b.x; }

template <class T>
T square(Point<T> p) { return dot(p,p); }

template <class T>
double length(Point<T> p) { return sqrt(double(square(p))); }

long double length(Point<long double> p) { return sqrt(square(p)); }
```

Line

```
template <class T>
struct Line
{
    Point<T> a,b;
    Line(Point<T> a_=Point<T>(),Point<T> b_=Point<T>()):a(a_),b(b_) {}
};
```

距离

```
template <class T>
double dis_PP(Point<T> a,Point<T> b) { return length(a-b); }

template <class T>
double dis_PL(Point<T> a,Line<T> l) { return fabs(cross(a-l.a,a-l.b))/dis_PP(l.a,l.b); }

template <class T>
double dis_PS(Point<T> a,Line<T> l)
{
    if (dot(a-l.a,l.b-l.a)<0) return dis_PP(a,l.a);
    if (dot(a-l.b,l.a-l.b)<0) return dis_PP(a,l.b);
    return dis_PL(a,l);
}
```

点绕中心旋转

```
template <class T>
Point<T> rotate(Point<T> a,double alpha)
{ return Point<T>(a.x*cos(alpha)-a.y*sin(alpha),a.x*sin(alpha)+a.y*cos(alpha)); }
```

关于线的对称点

```
template <class T>
Point<T> lineRoot(Point<T> a,Line<T> l)
{
    Point<T> v=l.b-l.a;
    return l.a+v*(dot(a-l.a,v)/dot(v,v));
}

template <class T>
Point<T> symmetry_PL(Point<T> a,Line<T> l) { return a+(lineRoot(a,l)-a)*2; }
```

位置关系判断

```
template <class T>
bool pointOnSegment(Point<T> a,Line<T> l)
{ return (sgn(cross(a-l.a,a-l.b))==0)&&(sgn(dot(a-l.a,a-l.b))<=0); }

template <class T>
bool lineCrossLine(Line<T> a,Line<T> b)
{
    double f1=cross(b.a-a.a,a.b-a.a),f2=cross(b.b-a.a,a.b-a.a);
    double g1=cross(a.a-b.a,b.b-b.a),g2=cross(a.b-b.a,b.b-b.a);
    return ((f1<0)^(f2<0))&&((g1<0)^(g2<0));
}

template <class T>
bool pointOnLineLeft(Point<T> a,Line<T> l) { return cross(l.b-l.a,a-l.a)>0; }

//适用任意多边形,O(n)
template <class T>
bool pointInPolygon(Point<T> a,const vector<Point<T>> &p)
{
    int n=p.size();
    for (int i=0;i<n;i++)
        if (pointOnSegment(a,Line<T>(p[i],p[(i+1)%n])))
            return 1;
    bool t=0;
    for (int i=0;i<n;i++)
    {
        Point<T> u=p[i],v=p[(i+1)%n];
        if (u.x<a.x&&v.x>=a.x&&pointOnLineLeft(a,Line<T>(v,u))) t^=1;
        if (u.x>=a.x&&v.x<a.x&&pointOnLineLeft(a,Line<T>(u,v))) t^=1;
    }
    return t;
}

//适用凸多边形,O(log n)
template <class T>
bool pointInPolygon_(Point<T> a,const vector<Point<T>> &p)
{
    int n=p.size();
    if (cross(a-p[0],p[1]-p[0])<0||cross(a-p[0],p[n-1]-p[0])>0) return 0;
    if (pointOnSegment(a,Line<T>(p[0],p[1]))||pointOnSegment(a,Line<T>(p[n-1],p[0]))) return 1;
    int l=1,r=n-1;
    while (l+1<r)
    {
        int mid=(l+r)>>1;
        if (cross(a-p[1],p[mid]-p[1])<0) l=mid;
        else r=mid;
    }
    if (cross(a-p[l],p[r]-p[l])>0) return 0;
    if (pointOnSegment(a,Line<T>(p[l],p[r]))) return 1;
    return 1;
}
```

线段交点

```
//小 心 平 行
template <class T>
```

```cpp
Point<T> lineIntersection(Line<T> a,Line<T> b)
{
    Point<T> u=a.a-b.a,v=a.b-a.a,w=b.b-b.a;
    double t=cross(u,w)/cross(w,v);
    return a.a+t*v;
}
```

过定点做圆的切线

```cpp
template <class T>
vector<Line<T>> tan_PC(Point<T> a,Point<T> c,T r)
{
    Point<T> v=c-a;
    vector<Line<T>> res;
    int dis=dis_PP(a,c);
    if (sgn(dis-r)==0) res.push_back(rotate(v,acos(-1)/2));
    else if (dis>r)
    {
        double alpha=asin(r/dis);
        res.push_back(rotate(v,alpha));
        res.push_back(rotate(v,-alpha));
    }
    return res;
}
```

两圆交点

```cpp
template <class T>
vector<Point<T>> circleIntersection(Point<T> c1,T r1,Point<T> c2,T r2)
{
    auto get=[&](Point<T> c,T r,double alpha)->Point<T>
    { return Point<T>(c.x+cos(alpha)*r,c.y+sin(alpha)*r); };

    auto angle=[&](Point<T> a)->double { return atan2(a.x,a.y); };

    vector<Point<T>> res;
    double d=dis_PP(c1,c2);
    if (sgn(d)==0) return res;
    if (sgn(r1+r2-d)<0) return res;
    if (sgn(fabs(r1-r2)-d)>0) return res;
    double alpha=angle(c2-c1);
    double beta=acos((r1*r1-r2*r2+d*d)/(r1*d*2));
    Point<T> p1=get(c1,r1,alpha-beta),p2=get(c1,r1,alpha+beta);
    res.push_back(p1);
    if (p1!=p2) res.push_back(p2);
    return res;
}
```

多边形面积

```cpp
template <class T>
double polygonArea(const vector<Point<T>> &p)
{
    int n=p.size();
    double res=0;
    for (int i=1;i<n-1;i++) res+=cross(p[i]-p[0],p[i+1]-p[0]);
    return fabs(res/2);
}
```

自适应辛普森法

```cpp
//注意边界函数值不能小于 eps
double f(double x) { return pow(x,0.5); }
double calc(double l,double r)
{
    double mid=(l+r)/2.0;
    return (r-l)*(f(l)+f(r)+f(mid)*4.0)/6.0;
}
double simpson(double l,double r,double lst)
```

```
{
    double mid=(l+r)/2.0;
    double fl=calc(l,mid),fr=calc(mid,r);
    if (sgn(fl+fr-lst)==0) return fl+fr;
    else return simpson(l,mid,fl)+simpson(mid,r,fr);
}
```

静态凸包

```
template <class T>
vector<Point<T>> getHull(vector<Point<T>> p)
{
    vector<Point<T>> h,l;
    sort(p.begin(),p.end(),[&](auto a,auto b)
    {
        if (a.x!=b.x) return a.x<b.x;
        else return a.y<b.y;
    });
    p.erase(unique(p.begin(),p.end()),p.end());
    if (p.size()<=1) return p;
    for (auto a:p)
    {
        while (h.size()>1&&sgn(cross(a-h.back(),a-h[h.size()-2]))<=0) h.pop_back();
        while (l.size()>1&&sgn(cross(a-l.back(),a-l[l.size()-2]))>=0) l.pop_back();
        l.push_back(a);
        h.push_back(a);
    }
    l.pop_back();
    reverse(h.begin(),h.end());
    h.pop_back();
    l.insert(l.end(),h.begin(),h.end());
    return l;
}
```

旋转卡壳求直径

```
template <class T>
double getDiameter(vector<Point<T>> p)
{
    double res=0;
    if (p.size()==2) return dis_PP(p[0],p[1]);
    int n=p.size();
    p.push_back(p.front());
    int j=2;
    for (int i=0;i<n;i++)
    {
        while (sgn(cross(p[i+1]-p[i],p[j]-p[i])-cross(p[i+1]-p[i],p[j+1]-p[i]))<0)
            j=(j+1)%n;
        res=max({res,dis_PP(p[i],p[j]),dis_PP(p[i+1],p[j])});
    }
    return res;
}
```

半平面交

```
template <class T>
vector<Point<T>> hp(vector<Line<T>> lines)
{
    sort(lines.begin(),lines.end(),[&](auto l1,auto l2)
    {
        auto d1=l1.b-l1.a;
        auto d2=l2.b-l2.a;

        if (sgn(d1)!=sgn(d2)) return sgn(d1)==1;
        return cross(d1,d2)>0;
    });

    deque<Line<T>> ls;
    deque<Point<T>> ps;
```

```
    for (auto l:lines)
    {
        if (ls.empty())
        {
            ls.push_back(l);
            continue;
        }
        while (!ps.empty()&&!pointOnLineLeft(ps.back(),l))
        {
            ps.pop_back();
            ls.pop_back();
        }
        while (!ps.empty()&&!pointOnLineLeft(ps[0],l))
        {
            ps.pop_front();
            ls.pop_front();
        }
        if (cross(l.b-l.a,ls.back().b-ls.back().a)==0)
        {
            if (dot(l.b-l.a,ls.back().b-ls.back().a)>0)
            {
                if (!pointOnLineLeft(ls.back().a,l))
                {
                    assert(ls.size()==1);
                    ls[0]=l;
                }
                continue;
            }
            return {};
        }
        ps.push_back(lineIntersection(ls.back(),l));
        ls.push_back(l);
    }
    while (!ps.empty()&&!pointOnLineLeft(ps.back(),ls[0]))
    {
        ps.pop_back();
        ls.pop_back();
    }
    if (ls.size()<=2) return {};
    ps.push_back(lineIntersection(ls[0],ls.back()));
    return vector(ps.begin(),ps.end());
}
```