Summary

Run summarize.bat to update this file.

Last Updated: 2025-09-07

Author: HuangZy

[TOC]

DataStructure

BigInt.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int, int>;
using i128 = __int128_t;
using pll = pair<ll, ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

struct BigInt {
    vector<int> a;
    BigInt(int x = 0) {
        if (x == 0) {
            a.push_back(0);
            return;
        }
        while (x) {
            a.push_back(x % 10);
            x /= 10;
        }
    }
    void update() {
        for (int i = 0;i<a.size();i++) {
            if (a[i] >= 10) {
                if (i + 1 < a.size()) {
                    a[i + 1] += a[i] / 10;
                    a[i] %= 10;
                } else {
                    a.push_back(a[i] / 10);
                    a[i] %= 10;
                }
            }
        }
        while (a.size() > 1 && a.back() == 0) {
```

```
            a.pop_back();
        }
        return;
    }
    BigInt operator*(const BigInt& A) const {
        BigInt B;
        B.a.resize(a.size() + A.a.size());
        for (int i = 0;i<a.size();i++) {
            for (int j = 0;j<A.a.size();j++) {
                B.a[i + j] += a[i] * A.a[j];
            }
        }
        B.update();
        return B;
    }
};

ostream &operator<<(ostream &o, const BigInt &a) {
    for (int i = a.a.size() - 1;i>=0;i--) {
        o << a.a[i];
    }
    return o;
}"\n"
```

CartesianTree.cpp

```
#include <bits/stdc++.h>
using namespace std;

/*
先 init 初始化
传入 vector<T> 构建小根堆笛卡尔树
*/

template<typename T>
class CartesianTree {
public:
    struct Node {
        T val;
        int ch[2];
        Node() {
            val = ch[0] = ch[1] = 0;
        }
    };
    vector<Node> nodes;

    void init(int N_) {
        nodes.clear();
        nodes.resize(N_ + 1);
    }
```

```
    void buildMinHeap(vector<T>& a) {
        vector<int> stk;
        stk.push_back(0);
        nodes[0].val = -INF;
        for (int i = 1;i<a.size();i++) {
            int lst = -1;
            while (a[stk.back()] > a[i]) {
                lst = stk.back();
                stk.pop_back();
            }

            if (lst != -1) {
                nodes[i].ch[0] = lst;
            }
            nodes[i].val = a[i];
            nodes[stk.back()].ch[1] = i;
            stk.push_back(i);
        }
        return;
    }
};"\n"
```

ChthollyTree.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;


// 注意下标别搞错了，什么从 0 开始之类...
struct ChthollyTree {
    struct Node {
        int l, r;
        int v;
        Node(int L_ = 0, int R_ = 0, int V_ = 0) {l = L_, r = R_, v = V_;}
        bool operator<(const Node& A) const {return l < A.l;}
    };
    set<Node> nodes;

    // 注意初始化插入全 1 段.
    // 注意插入 n + 1
    ChthollyTree(int N_) {
        nodes.insert(Node(1, N_ + 1, 0));
    }
```

```
    set<Node>::iterator split(int x) {
        auto it = nodes.lower_bound(Node(x, 0));
        if (it != nodes.end() && it->l == x) return it;
        -- it;
        int l = it->l, r = it->r, v = it->v;
        nodes.erase(it);
        nodes.insert(Node(l, x - 1, v));
        return nodes.insert(Node(x, r, v)).first;
    }

    void assign(int l, int r, int v) {
        auto itr = split(r + 1), itl = split(l);
        nodes.erase(itl, itr);
        nodes.insert(Node(l, r, v));
        return;
    }
};

struct ChthollyTree {
    const int M = 1e8;
    struct Node {
        int l,r;
        bool operator<(const Node& A) const {
            if (l != A.l) return l < A.l;
            return r < A.r;
        }
        Node (int L_ = 0,int R_ = 0) {
            l = L_, r = R_;
        }
    };
    set<Node> nodes;

    void init() {
        nodes.insert(Node(1,M));
    }

    void insert(Node a) {
        Node b; b.l = a.l, b.r = M;
        auto it = nodes.upper_bound(b);

        vector<Node> val;

        if (it == nodes.end()) {
            it = prev(nodes.end());
            if (it->r > a.r) {
                if (it->l <= a.l - 1) {
                    val.push_back(Node(it->l,a.l-1));
                }
                if (it->r >= a.r + 1) {
                    val.push_back(Node(a.r+1,it->r));
                }
                it = nodes.erase(it);
            } else if (it->r >= a.l) {
                if (it->l <= a.l - 1) {
```

```
                    val.push_back(Node(it->l,a.l-1));
                }
                it = nodes.erase(it);
            } else {
                ++ it;
            }
        } else {
            if (it == nodes.begin()) {

            } else {
                -- it;
                if (it->r > a.r) {
                    if (it->l <= a.l - 1) {
                        val.push_back(Node(it->l,a.l-1));
                    }
                    if (it->r >= a.r + 1) {
                        val.push_back(Node(a.r+1,it->r));
                    }
                    it = nodes.erase(it);
                } else if (it->r >= a.l) {
                    if (it->l <= a.l - 1) {
                        val.push_back(Node(it->l,a.l-1));
                    }
                    it = nodes.erase(it);
                } else {
                    ++ it;
                }
            }
        }

        while (it != nodes.end() && it->l <= a.r) {
            if (it->r <= a.r) {
                it = nodes.erase(it);
            } else {
                if (it->r >= a.r + 1) {
                    val.push_back(Node(a.r+1,it->r));
                }
                it = nodes.erase(it);
            }
        }

        nodes.insert(a);
        for (auto v : val) {
            nodes.insert(v);
        }
        return;
    }
} ;
"\n"
```

Dsu.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;

struct Dsu{
    int fa[N];
    void init(int n){
        for (int i = 1;i<n+1;i++){
            fa[i] = i;
        }
    }
    int find(int x){
        if (x == fa[x]){
            return x;
        }
        return fa[x] = find(fa[x]);
    }
    void merge(int x,int y){
        x = find(x),y = find(y);
        if (x == y){
            return;
        }
        fa[x] = y;
        return;
    }
};"\n"
```

DsuOnTree.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using i128 = __int128_t;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

struct DsuOnTree {
    vector<vector<int>> g;
    int n,cntDfn;
    vector<int> dfn,L,R,son,siz;
    // 先填充 g
    void init() {
        cntDfn = 0;
        dfn.resize(n + 1,0);
        L.resize(n + 1,0);
```

```
        R.resize(n + 1,0);
        son.resize(n + 1,0);
        siz.resize(n + 1,0);
        g.resize(n + 1);
        return;
    }

    void dfsPre (int u,int f) {
        dfn[u] = ++cntDfn;
        L[u] = dfn[u];
        siz[u] = 1;
        for (auto v : g[u]) {
            if (v == f) continue;
            dfsPre(v,u);
            siz[u] += siz[v];
            if (!son[u] || siz[son[u]] < siz[v]) {
                son[u] = v;
            }
        }
        R[u] = cntDfn;
        return;
    };

    void add(int x) {
        return;
    }

    void del(int x) {
        return;
    }

    void dfs(int u,int f,bool keep) {
        for (auto v : g[u]) {
            if (v == f || v == son[u]) {
                continue;
            }
            dfs(v,u,0);
        }

        if (son[u]) dfs(son[u],u,1);

        for (auto v : g[u]) {
            if (v == f || v == son[u]) {
                continue;
            }
            for (int i = L[v];i<=R[v];i++) {
                // add();
            }
        }
        // add();

        if (!keep) {
            for (int i = L[u];i<=R[u];i++) {
                // del();
```

```
            }
        }
        return;
    };

};"\n"
```

FenwickTree.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;

struct BIT {
    int n;
    struct Info {
        Info() {

        }
        Info operator+(const Info& A) const {

        }
        Info operator-(const Info& A) const {

        }
    };
    vector<Info> infos;
    void init(int N_) {
        n = N_ + 1;
        infos.assign(n + 1, Info());
    }
    int lowbit(int x) {return x & -x;}
    void update(int x, ll v) {
        x ++;
        while (x <= n) {

            x += lowbit(x);
        }
    }
    Info query(int x) {
        x ++;
        Info res;
        while (x) {
            res = res + infos[x];
            x -= lowbit(x);
        }
        return res;
    }
    Info query(int l, int r) {
        if (l > r) return Info();
```

```
        return query(r) - query(l - 1);
    }
};
"\n"
```

LeftleaningHeap.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;
class LeftLeaningHeap{
public:
    #define ls(x) nodes[x].ls
    #define rs(x) nodes[x].rs
    struct Node{
        int fa,dist,ls,rs,val,laz;
        bool operator<(const Node& A) const{
            return val > A.val;
        }
    } nodes[N];

    LeftLeaningHeap(){
        nodes[0].dist = -1;
    }

    int find(int x){
        if (x == nodes[x].fa){
            return x;
        }
        return nodes[x].fa = find(nodes[x].fa);
    }

    void pushUp(int x){
        nodes[x].dist = nodes[ls(x)].dist + 1;
        nodes[ls(x)].fa = nodes[rs(x)].fa = x;
    }

    int merge(int x,int y) {
        if (!x || !y) return x | y;

        if (nodes[x] < nodes[y]) {
            swap(x,y);
        }

        nodes[x].rs = merge(nodes[x].rs,y);
        if (nodes[ls(x)].dist < nodes[rs(x)].dist){
            swap(ls(x),rs(x));
        }

        pushUp(x);
```

```
        return x;
    }

    void erase(int x){
        nodes[x].fa = nodes[ls(x)].fa = nodes[rs(x)].fa = merge(ls(x),rs(x));
        nodes[x].dist = nodes[x].ls = nodes[x].rs = 0;
    }
};"\n"
```

LiChaoTree.cpp

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const ll INF = 1e18;
/*
查最小值就把整棵树按照 x 轴翻转
先用 init 函数初始化
注意空间开到 4 倍!!!

使用方法是先 add 再 update
*/
class LiChaoTree {
public:
    typedef pair<double, int> pdi;

    const double eps = 1e-9;
    const double NINF = -1e18;
    int cmp(double x, double y) {
        if (x - y > eps) return 1;
        if (y - x > eps) return -1;
        return 0;
    }

    struct line {
        double k, b;
    } p[100005];

    int s[160005];
    int cnt;

    void init(int x) {
        cnt = 0;
        p[0].k = 0; p[0].b = NINF;
        fill(s, s + x * 4 + 10, 0);
    }


    double calc(int id, int d) { return p[id].b + p[id].k * d; }

    void add(int x0, int y0, int x1, int y1) {
```

```
        cnt++;
        if (x0 == x1)  // 特判直线斜率不存在的情况
            p[cnt].k = 0, p[cnt].b = max(y0, y1);
        else
            p[cnt].k = double(1) * (y1 - y0) / (x1 - x0), p[cnt].b = y0 - p[cnt].k
* x0;
    }

    void upd(int root, int cl, int cr, int u) {  // 对线段完全覆盖到的区间进行修改
        int &v = s[root], mid = (cl + cr) >> 1;
        int bmid = cmp(calc(u, mid), calc(v, mid));
        if (bmid == 1 || (!bmid && u < v)) swap(u, v);
        int bl = cmp(calc(u, cl), calc(v, cl)), br = cmp(calc(u, cr), calc(v,
cr));
        if (bl == 1 || (!bl && u < v)) upd(root << 1, cl, mid, u);
        if (br == 1 || (!br && u < v)) upd(root << 1 | 1, mid + 1, cr, u);
    }

    void update(int root, int cl, int cr, int l, int r,
                int u) {  // 定位插入线段完全覆盖到的区间
        if (l <= cl && cr <= r) {
            upd(root, cl, cr, u);
            return;
        }
        int mid = (cl + cr) >> 1;
        if (l <= mid) update(root << 1, cl, mid, l, r, u);
        if (mid < r) update(root << 1 | 1, mid + 1, cr, l, r, u);
    }

    pdi pmax(pdi x, pdi y) {  // pair max函数
        if (cmp(x.first, y.first) == -1)
            return y;
        else if (cmp(x.first, y.first) == 1)
            return x;
        else
            return x.second < y.second ? x : y;
    }

    pdi query(int root, int l, int r, int d) {  // 查询
        if (r < d || d < l) return {0, 0};
        int mid = (l + r) >> 1;
        double res = calc(s[root], d);
        if (l == r) return {res, s[root]};
        return pmax({res, s[root]}, pmax(query(root << 1, l, mid, d),
                            query(root << 1 | 1, mid + 1, r, d)));
    }
};

/*
当插入的线段 k 和 b 都为整数时,使用这个版本可以提高精度.

直接 addLine(k,b) 插入, query(x) 查询.
*/
struct LiChao {
```

```
    struct Line { ll m, b; };
    struct Node { Line ln; Node *l, *r;
        Node(Line v):ln(v),l(nullptr),r(nullptr){}
    };
    ll L, R;
    Node *root;
    LiChao(ll _L, ll _R):L(_L),R(_R),root(nullptr){}

    ll eval(const Line &ln, ll x) {
        return ln.m*x + ln.b;
    }

    void addLine(Line nw, Node *&nd, ll l, ll r) {
        if (!nd) { nd = new Node(nw); return; }
        ll m = (l + r) >> 1;
        bool lef = eval(nw, l) < eval(nd->ln, l);
        bool mid = eval(nw, m) < eval(nd->ln, m);
        if (mid) swap(nw, nd->ln);
        if (r - l == 0) return;
        if (lef != mid) addLine(nw, nd->l, l, m);
        else addLine(nw, nd->r, m+1, r);
    }

    void addLine(ll m, ll b) {
        addLine({m, b}, root, L, R);
    }

    ll query(ll x, Node *nd, ll l, ll r) {
        if (!nd) return INF;
        ll res = eval(nd->ln, x);
        if (l==r) return res;
        ll m = (l + r) >> 1;
        if (x <= m) return min(res, query(x, nd->l, l, m));
        else return min(res, query(x, nd->r, m+1, r));
    }

    ll query(ll x) {
        return query(x, root, L, R);
    }
};
"\n"
```

SegmentTree.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;
const ll INF = 1e18;

#define LS (rt << 1)
```

```
#define RS (rt << 1 | 1)

class SegTree{
public:
    struct Node{
        Node() {

        }
    }nodes[N*4];

    Node merge(Node L,Node R){
        Node M;
        return M;
    }
    void build(int rt,int l,int r){
        if (l == r){
            return;
        }
        int mid = l + r >> 1;
        build(LS,l,mid),build(RS,mid+1,r);
        nodes[rt] = merge(nodes[LS],nodes[RS]);
    }
    void pd(int rt){

    }
    void update(int rt,int l,int r,int ql,int qr,int val){
        if (ql > qr || l > qr || ql > r) return;
        if (ql <= l && r <= qr){
            return;
        }
        int mid = l+r>>1;
        pd(rt);
        if (ql <= mid){
            update(LS,l,mid,ql,qr,val);
        }
        if (qr >= mid + 1){
            update(RS,mid+1,r,ql,qr,val);
        }
        nodes[rt] = merge(nodes[LS],nodes[RS]);
        return;
    }
    void modify(int rt,int l,int r,int ql,int qr,int val){
        if (ql > qr || l > qr || ql > r) return;
        if (ql <= l && r <= qr){
            return;
        }
        int mid = l+r>>1;
        pd(rt);
        if (ql <= mid){
            modify(LS,l,mid,ql,qr,val);
        }
        if (qr >= mid + 1){
            modify(RS,mid+1,r,ql,qr,val);
        }
```

```
        nodes[rt] = merge(nodes[LS],nodes[RS]);
        return;
    }
    Node query(int rt,int l,int r,int ql,int qr){
        if (ql > qr) {
            return Node();
        }
        if (ql <= l && r <= qr){
            return nodes[rt];
        }
        int mid = l+r>>1;
        pd(rt);
        if (ql > mid){
            return query(RS,mid+1,r,ql,qr);
        }else if (qr < mid + 1){
            return query(LS,l,mid,ql,qr);
        }else{
            return merge(query(LS,l,mid,ql,qr),query(RS,mid+1,r,ql,qr));
        }
    }
};

class SegTreePlusMul{
struct Node{
    ll sum,plus,mul,len;
};
private:
    Node nodes[N];
public:
    ll mod;
    Node merge(Node A,Node B){
        Node C;
        C.plus = 0;
        C.mul = 1;
        C.sum = (A.sum + B.sum) % mod;
        C.len = A.len + B.len;
        return C;
    }
    void build(ll rt,ll l,ll r){
        if (l==r){
            cin >> nodes[rt].sum;
            nodes[rt].len = 1;
            nodes[rt].plus = 0;
            nodes[rt].mul = 1;
            return;
        }
        ll mid = l+r>>1;
        build(LS,l,mid),build(RS,mid+1,r);
        nodes[rt] = merge(nodes[LS],nodes[RS]);
    }
    void pushDown(int rt){
        nodes[RS].sum =
(nodes[RS].sum*nodes[rt].mul%mod+nodes[rt].plus*nodes[RS].len%mod) % mod;
        nodes[LS].sum =
```

```
(nodes[RS].sum*nodes[rt].mul%mod+nodes[rt].plus*nodes[LS].len%mod) % mod;

        nodes[RS].mul = (nodes[RS].mul*nodes[rt].mul) % mod;
        nodes[LS].mul = (nodes[LS].mul*nodes[rt].mul) % mod;

        nodes[RS].plus = (nodes[RS].plus*nodes[rt].mul+nodes[rt].plus) % mod;
        nodes[LS].plus = (nodes[LS].plus*nodes[rt].mul+nodes[rt].plus) % mod;

        nodes[rt].plus = 0;
        nodes[rt].mul = 1;
    }
    void update(ll rt,ll l,ll r,ll ql,ll qr,ll mode,ll k){
        if (ql <= l && r <= qr){
            if (mode == 1){
                nodes[rt].plus += k;
                nodes[rt].plus %= mod;
                nodes[rt].sum += (k * nodes[rt].len) % mod;
                nodes[rt].sum %= mod;
            }else{
                nodes[rt].plus *= k;
                nodes[rt].plus %= mod;
                nodes[rt].mul *= k;
                nodes[rt].mul %= mod;
                nodes[rt].sum *= k;
                nodes[rt].sum %= mod;
            }
            return;
        }
        pushDown(rt);
        ll mid = l+r>>1;
        if (ql<=mid){
            update(LS,l,mid,ql,qr,mode,k);
        }
        if (qr >= mid+1){
            update(RS,mid+1,r,ql,qr,mode,k);
        }
        nodes[rt] = merge(nodes[LS],nodes[RS]);
    }
    Node query(ll rt,ll l,ll r,ll ql,ll qr){
        if (ql <= l && r <= qr){
            return nodes[rt];
        }
        pushDown(rt);
        ll mid = l+r>>1;
        if (qr<=mid){
            return query(LS,l,mid,ql,qr);
        }else if (ql>=mid+1){
            return query(RS,mid+1,r,ql,qr);
        }else{
            return merge(query(LS,l,mid,ql,qr),query(RS,mid+1,r,ql,qr));
        }
    }
};
```

```
// 调用前先 init
#define LS nodes[rt].ls
#define RS nodes[rt].rs
struct DynamicSegTree {
    struct Node {
        int ls,rs;
        Node() {
            ls = rs = 0;
        }
    } nodes[N];
    int cntNode;
    void init() {
        cntNode = 1;
        nodes[0] = Node();
    }
    int newNode() {
        nodes[cntNode] = Node();
        return cntNode++;
    }
    Node merge(Node L,Node R) {
        Node M;
        // 补充合并方法
        return M;
    }
    void pushUp(int rt) {
        // 更新父节点
        return;
    }
    void pushDown(int rt) {
        // 下传标记
    }
    void update(int& rt,int l,int r,int ql,int qr,int val) {
        if (ql > qr || l > qr || r < ql) return;
        if (!rt) rt = newNode();
        if (ql <= l && r <= qr) {
            // 更新方法
            return;
        }
        int mid = l + r >> 1;
        update(nodes[rt].ls,l,mid,ql,qr,val);
        update(nodes[rt].rs,mid+1,r,ql,qr,val);
        pushUp(rt);
        return;
    }
    Node query(int& rt,int l,int r,int ql,int qr) {
        if (ql > qr || l > qr || r < ql || !rt) return Node();
        if (ql <= l && r <= qr) {
            return nodes[rt];
        }
        int mid = l + r >> 1;
        pushDown(rt);
        return merge(query(LS,l,mid,ql,qr),query(RS,mid+1,r,ql,qr));
    }
```

```
    int treeMerge(int& ls,int& rs,int l,int r) {
        if (!ls || !rs) return ls | rs;
        if (l == r) {
            // 叶节点合并方法

            return ls;
        }
        int mid = l + r >> 1;
        nodes[ls].ls = treeMerge(nodes[ls].ls,nodes[rs].ls,l,mid);
        nodes[ls].rs = treeMerge(nodes[ls].rs,nodes[rs].rs,mid+1,r);
        pushUp(ls);
        return ls;
    }
};


struct DynamicSegTree {
    #define LS info[rt].ls
    #define RS info[rt].rs
    struct Info {
        int ls, rs;
        Info() {
            ls = rs = 0;
        }
        Info operator+(const Info& A) const {

        }
    } info[N];
    int cntNode = 0;

    void init() {
        cntNode = 0;
    };

    int newNode() {
        ++cntNode;
        info[cntNode] = Info();
        return cntNode;
    };

    void pushUp(int rt) {
        return;
    }

    void build(int& rt,int l,int r) {
        rt = newNode();
        if (l == r) {
            return;
        }
        int mid = l + r >> 1;
        build(LS,l,mid); build(RS,mid+1,r);
        pushUp(rt);
        return;
    }
```

```
// 主席树版本
// void update(int& rt,int l,int r,int ql,int qr) {
//  if (ql > qr || qr < l || r < ql) return;
//  if (ql <= l && r <= qr) {
//      return;
//  }
//  int mid = l + r >> 1;
//  if (ql <= mid) {
//      ++cntNode;
//      info[cntNode] = info[LS];
//      LS = cntNode;
//      update(LS,l,mid,ql,qr);
//  }
//  if (qr >= mid + 1) {
//      ++cntNode;
//      info[cntNode] = info[RS];
//      RS = cntNode;
//      update(RS,mid+1,r,ql,qr);
//  }
//  pushUp(rt);
//  return;
// }

void update(int& rt,int l,int r,int ql,int qr) {
    if (ql > qr || qr < l || r < ql) return;
    if (!rt) rt = newNode();
    if (ql <= l && r <= qr) {
        return;
    }
    int mid = l + r >> 1;
    if (ql <= mid) {
        update(LS,l,mid,ql,qr);
    }
    if (qr >= mid + 1) {
        update(RS,mid+1,r,ql,qr);
    }
    pushUp(rt);
    return;
}

Info query(int rt,int l,int r,int ql,int qr) {
    if (!rt || ql > qr || r < ql || qr < l) return Info();
    if (ql <= l && r <= qr) {
        return info[rt];
    }
    int mid = l + r >> 1;
    if (qr <= mid) {
        return query(LS,l,mid,ql,qr);
    } else if (ql >= mid + 1) {
        return query(RS,mid+1,r,ql,qr);
    } else {
        return query(LS,l,mid,ql,qr) + query(RS,mid+1,r,ql,qr);
    }
```

```
    }
} solver;

"\n"
```

Splay.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;

/*
支持自定义结构体从小到大排序的平衡树
注意在使用 suc 或者 pre 时,如果树中不存在这个前后驱,会 re
可通过插入正负无穷解决这个问题
*/

template<typename T, typename Compare = std::less<T>>
class SplayTree {
private:
    int siz;
    struct Node {
        T key;
        Node *left, *right, *parent;
        int cnt;
        std::size_t sz;
        Node(const T& k, Node* p=nullptr)
          : key(k), left(nullptr), right(nullptr), parent(p), cnt(1), sz(1) {}
    };

    Node* root = nullptr;
    Compare comp;

    void update(Node* x) {
        x->sz = x->cnt
              + (x->left ? x->left->sz : 0)
              + (x->right ? x->right->sz : 0);
    }

    void rotate_left(Node* x) {
        Node* y = x->right;
        x->right = y->left;
        if (y->left) y->left->parent = x;
        y->parent = x->parent;
        if (!x->parent) root = y;
        else if (x == x->parent->left) x->parent->left = y;
        else x->parent->right = y;
        y->left = x; x->parent = y;
        update(x); update(y);
    }
```

```
    void rotate_right(Node* x) {
        Node* y = x->left;
        x->left = y->right;
        if (y->right) y->right->parent = x;
        y->parent = x->parent;
        if (!x->parent) root = y;
        else if (x == x->parent->left) x->parent->left = y;
        else x->parent->right = y;
        y->right = x; x->parent = y;
        update(x); update(y);
    }

    void splay(Node* x) {
        while (x->parent) {
            Node* p = x->parent;
            Node* g = p->parent;
            if (!g) {
                if (x == p->left) rotate_right(p);
                else rotate_left(p);
            } else if ((x == p->left) == (p == g->left)) {
                if (x == p->left) { rotate_right(g); rotate_right(p); }
                else { rotate_left(g); rotate_left(p); }
            } else {
                if (x == p->left) { rotate_right(p); rotate_left(g); }
                else { rotate_left(p); rotate_right(g); }
            }
        }
    }

    Node* find_node(const T& key) {
        Node* x = root;
        while (x) {
            if (comp(key, x->key)) x = x->left;
            else if (comp(x->key, key)) x = x->right;
            else return x;
        }
        return nullptr;
    }

public:
    SplayTree(Compare c = Compare()) : comp(c) {}

    void insert(const T& key) {
        siz++;
        if (!root) {
            root = new Node(key);
            return;
        }
        Node* x = root;
        Node* p = nullptr;
        while (x && !( !comp(key, x->key) && !comp(x->key, key) )) {
            p = x;
            x = comp(key, x->key) ? x->left : x->right;
```

```
        }
        if (x) {
            x->cnt++;
            splay(x);
        } else {
            x = new Node(key, p);
            if (comp(key, p->key)) p->left = x;
            else p->right = x;
            splay(x);
        }
    }

    bool contains(const T& key) {
        Node* x = find_node(key);
        if (x) { splay(x); return true; }
        return false;
    }

    void erase(const T& key) {
        Node* x = find_node(key);
        if (!x) return;
        siz--;
        splay(x);
        if (x->cnt > 1) {
            x->cnt--;
            update(x);
            return;
        }
        Node* L = x->left;
        Node* R = x->right;
        delete x;
        if (L) L->parent = nullptr;
        if (!L) {
            root = R;
            if (R) R->parent = nullptr;
        } else {
            Node* m = L;
            while (m->right) m = m->right;
            splay(m);
            m->right = R;
            if (R) R->parent = m;
            root = m;
            update(root);
        }
    }

    bool empty() const { return root == nullptr; }

    std::size_t size() const { return root ? root->sz : 0; }

    T kth(std::size_t k) {
        if (!root || k<1 || k>root->sz) throw std::out_of_range("k");
        Node* x = root;
        while (x) {
```

```
            std::size_t L = x->left? x->left->sz : 0;
            if (k <= L) x = x->left;
            else if (k > L + x->cnt) {
                k -= L + x->cnt;
                x = x->right;
            } else {
                splay(x);
                return x->key;
            }
        }
        throw std::out_of_range("k");
    }

    std::size_t getRank(const T& key) {
        if (!root) return 0;
        Node* x = root;
        std::size_t rank = 0;
        while (x) {
            if (comp(key, x->key)) {
                x = x->left;
            } else {
                std::size_t L = x->left? x->left->sz : 0;
                rank += L;
                if (!comp(x->key, key) && !comp(key, x->key)) {
                    splay(x);
                    return rank;
                }
                rank += x->cnt;
                x = x->right;
            }
        }
        if (x) splay(x);
        return rank;
    }

    T pre(const T& key) {
        Node* x = root;
        Node* pred = nullptr;
        while (x) {
            if (comp(x->key, key)) {
                pred = x;
                x = x->right;
            } else x = x->left;
        }
        if (!pred) throw std::out_of_range("no predecessor");
        splay(pred);
        return pred->key;
    }

    T suc(const T& key) {
        Node* x = root;
        Node* succ = nullptr;
        while (x) {
            if (comp(key, x->key)) {
```

```
                succ = x;
                x = x->left;
            } else x = x->right;
        }
        if (!succ) throw std::out_of_range("no successor");
        splay(succ);
        return succ->key;
    }

    int size() {
        return siz;
    }
};"\n"
```

Trie.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;


struct Trie {
    int cnt;
    struct Node {
        int ch[2];
        Node () {
            ch[0] = ch[1] = -1;
        }
    };
    vector<Node> nodes;

    int newNode() {
        nodes.push_back(Node());
        return ++cnt;
    }

    Trie() {
        nodes.resize(1);
        cnt = 0;
    };

    void insert(ll x) {
        int rt = 0;
        for (int i = 30;i>=0;i--) {
            int nxt;
```

```
            if ((x>>i)&1) {
                nxt = 1;
            } else {
                nxt = 0;
            }

            if (nodes[rt].ch[nxt] == -1) {
                nodes[rt].ch[nxt] = newNode();
            }

            rt = nodes[rt].ch[nxt];
        }
        return;
    }

    ll getMx(ll x) {
        int rt = 0;
        ll ans = 0;
        for (int i = 30;i>=0;i--) {
            int nxt;
            if ((x>>i)&1) {
                nxt = 0;
            } else {
                nxt = 1;
            }

            if (nodes[rt].ch[nxt] == -1) {
                if (nodes[rt].ch[nxt^1] == -1) {
                    return 0;
                } else {
                    rt = nodes[rt].ch[nxt^1];
                }
            } else {
                rt = nodes[rt].ch[nxt];
                ans = ans + (1<<i);
            }
        }
        return ans;
    }

    void init() {
        cnt = 0;
        nodes.clear();
        nodes.resize(1);
    }
};"\n"
```

ConvexHull.cpp

```
#include <bits/stdc++.h>
using namespace std;
```

```cpp
#define int long long
typedef long long ll;
const ll N = 2000000;

/*
使用说明:
手动输入点的数量n,各个点的坐标,然后init即可找出凸包.
可以处理重点,不会出现三点共线的问题.
默认double.
使用极角排序的方式寻找凸包.
*/

struct Point {
    double x, y;
    int idx;
    Point(double X = 0, double Y = 0) {
        x = X;
        y = Y;
    }

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

    Point operator+(const Point& A) const{
        return Point(x + A.x,y + A.y);
    }

    double operator^(const Point& A) const {
        return x * A.y - A.x * y;
    }

    double operator*(const Point& A) const {
        return x * A.x + y * A.y;
    }

    double len() {
        return sqrt(x * x + y * y);
    }

    double len2() {
        return x * x + y * y;
    }

    bool operator==(const Point& A) const {
        return (x == A.x) && (y == A.y);
    }
} ;

class ConvexHull {
public:
    int n;
    bool vis[N];
    Point points[N], hull[N];
```

```
    static bool cmp_y(const Point& A, const Point& B) {
        if (A.y == B.y){
            return A.x < B.x;
        }
        return A.y < B.y;
    }

    static bool cmp_sita(const Point& A, const Point& B, const Point& base) {
        Point A_base = A - base;
        Point B_base = B - base;
        if ((A_base ^ B_base) == 0) {
            return A_base.len() > B_base.len();
        }
        return (A_base ^ B_base) < 0;
    }

    int tp;

    void init() {
        tp = 1;
        sort(points + 1, points + 1 + n, cmp_y);
        hull[1] = points[1];
        sort(points + 2, points + 1 + n, [&base = hull[1]](const Point& A, const
Point& B) { return cmp_sita(A, B, base); });
        int cur = 2;
        for (; cur <= n; cur++) {
            if (points[cur] == hull[1]) continue;
            else { hull[++tp] = points[cur]; break;}
        }
        for (cur++; cur <= n; cur++) {
            if (hull[tp] == points[cur]) continue;
            Point L = hull[tp] - hull[tp - 1];
            Point R = points[cur] - hull[tp];
            if (R.x == 0 && R.y == 0) continue;
            if ((L ^ R) > 0) {
                while ((L ^ R) >= 0 && tp > 1){
                    if ((L ^ R) == 0){
                        if ((L * R) < 0){
                            break;
                        }else{
                            --tp;break;
                        }
                    }
                    tp--;
                    L = hull[tp] - hull[tp - 1];
                    R = points[cur] - hull[tp];
                }
                hull[++tp] = points[cur];
            } else if ((L ^ R) < 0) {
                hull[++tp] = points[cur];
            } else {
                if ((L * R) < 0) {
                    continue;
                } else {
```

```
                    hull[tp] = points[cur];
                }
            }
        }
    }
} ;"\n"
```

Triangle.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;

/*
使用说明:
手动输入Triangle中三个点的坐标
update函数更新外心左边和外接圆半径
可以应对三点共线的特殊情况
*/

struct Point {
    double x, y;
    int idx;
    Point(double X = 0, double Y = 0) {
        x = X;
        y = Y;
    }

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

    Point operator+(const Point& A) const{
        return Point(x + A.x,y + A.y);
    }

    double operator^(const Point& A) const {
        return x * A.y - A.x * y;
    }

    double operator*(const Point& A) const {
        return x * A.x + y * A.y;
    }

    double len() {
        return sqrt(x * x + y * y);
    }

    double len2() {
        return x * x + y * y;
```

```
    }

    bool operator==(const Point& A) const {
        return (x == A.x) && (y == A.y);
    }
};

class Triangle{
public:
    Point nodes[3];
    Point circumCenter;
    double r;
    void update(){
        double a,b,c,d,e,f;
        a=nodes[1].x-nodes[0].x,b=nodes[1].y-nodes[0].y,c=nodes[2].x-nodes[1].x;
        d=nodes[2].y-nodes[1].y;e=nodes[1].x*nodes[1].x+nodes[1].y*nodes[1].y
        -nodes[0].x*nodes[0].x-nodes[0].y*nodes[0].y;
        f=nodes[2].x*nodes[2].x+nodes[2].y*nodes[2].y-nodes[1].x*nodes[1].x
        -nodes[1].y*nodes[1].y;
        if (a*d == c*b){
            r = 0;
            if ((nodes[0] - nodes[1]).len() > r){
                r = (nodes[0] - nodes[1]).len();
                circumCenter = nodes[0] + nodes[1];
                circumCenter.x /= 2,circumCenter.y /= 2;
            }
            if ((nodes[0] - nodes[2]).len() > r){
                r = (nodes[0] - nodes[2]).len();
                circumCenter = nodes[0] + nodes[2];
                circumCenter.x /= 2,circumCenter.y /= 2;
            }
            if ((nodes[2] - nodes[1]).len() > r){
                r = (nodes[2] - nodes[1]).len();
                circumCenter = nodes[2] + nodes[1];
                circumCenter.x /= 2,circumCenter.y /= 2;
            }
            return;
        }
        circumCenter.x=(f*b-e*d)/(c*b-a*d)/2;
        circumCenter.y=(a*f-e*c)/(a*d-b*c)/2;
        r=(nodes[0]-circumCenter).len();
        return;
    }
};"\n"
```

CentroidTree.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
```

```cpp
using pii = pair<int,int>;
using i128 = __int128_t;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

// 传入一颗树，返回对应的点分树，已删去指向父亲的边!! 入度为 0 的就是根
vector<vector<int>> CentroidTree(vector<vector<int>>&g) {
    int n = g.size() - 1;
    vector<vector<int>> e(n + 1);
    vector<bool> bad(n + 1, 0);
    vector<int> siz(n + 1, 0);
    auto getCenter = [&](int u, int tot) -> int {
        int mn = n + 1, res;
        auto dfs = [&](auto&&self, int u, int f) -> void {
            int val = 1;
            siz[u] = 1;
            for (auto v : g[u]) {
                if (v == f || bad[v]) continue;
                self(self, v, u);
                siz[u] += siz[v];
                val = max(val, siz[v]);
            }
            val = max(val, tot - siz[u]);
            if (val < mn) {
                mn = val;
                res = u;
            }
            return;
        };
        dfs(dfs, u, u);
        return res;
    };
    auto getSiz = [&](auto&&self, int u, int f) -> int {
        int res = 1;
        for (auto v : g[u]) {
            if (v == f || bad[v]) continue;
            res += self(self, v, u);
        }
        return res;
    };
    vector<int> roots;
    int rt = getCenter(1, n);
    roots.push_back(rt);
    while (!roots.empty()) {
        rt = roots.back(); roots.pop_back();
        bad[rt] = 1;
        for (auto v : g[rt]) {
            if (bad[v]) continue;
            v = getCenter(v, getSiz(getSiz, v, v));
            e[rt].push_back(v);
            roots.push_back(v);
        }
```

```
    }
    return e;
};
"\n"
```

Dinic.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;
const int INF = 1e9;

class Graph {
public:
    int n;
    struct Edge {
        int next, to, rev;
        ll w;
    } edge[2 * N];
    int head[N];
    int cnt;

    void init(int N_) {
        n = N_;
        for (int i = 0; i <= n; i++)
            head[i] = 0;
        cnt = 0;
    }

    void add_edge(int u, int v, ll w) {
        cnt++;
        edge[cnt].to = v;
        edge[cnt].w = w;
        edge[cnt].next = head[u];
        head[u] = cnt;
        int forward_index = cnt;
        cnt++;
        edge[cnt].to = u;
        edge[cnt].w = 0;
        edge[cnt].next = head[v];
        head[v] = cnt;
        int reverse_index = cnt;
        edge[forward_index].rev = reverse_index;
        edge[reverse_index].rev = forward_index;
    }
} g;

class Dinic {
private:
    Graph* G;
```

```
    int s, t;
    vector<ll> d;
    vector<int> cur;

    bool bfs() {
        fill(d.begin(), d.end(), -1);
        queue<int> q;
        d[s] = 0;
        q.push(s);
        while (!q.empty()) {
            int u = q.front(); q.pop();
            for (int i = G->head[u]; i; i = G->edge[i].next) {
                int v = G->edge[i].to;
                if (d[v] < 0 && G->edge[i].w > 0) {
                    d[v] = d[u] + 1;
                    q.push(v);
                }
            }
        }
        return d[t] >= 0;
    }

    ll dfs(int u, ll flow) {
        if (u == t) return flow;
        for (int &i = cur[u]; i; i = G->edge[i].next) {
            int v = G->edge[i].to;
            if (d[v] == d[u] + 1 && G->edge[i].w > 0) {
                ll pushed = dfs(v, min(flow, G->edge[i].w));
                if (pushed) {
                    G->edge[i].w -= pushed;
                    G->edge[G->edge[i].rev].w += pushed;
                    return pushed;
                }
            }
        }
        return 0;
    }

public:
    Dinic(Graph* graph, int source, int sink) : G(graph), s(source), t(sink) {
        d.resize(G->n + 1);
        cur.resize(G->n + 1);
    }

    ll max_flow() {
        ll flow = 0;
        while (bfs()) {
            for (int i = 0; i <= G->n; i++)
                cur[i] = G->head[i];
            while (ll pushed = dfs(s, INF))
                flow += pushed;
        }
        return flow;
```

```
    }
};"\n"
```

Graph.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
const ll N = 2000000;

// 尤其注意初始化 Graph 类的时候边的数量要分配够,双向边开两倍!!!

template<typename T>
class Graph{
public:
    int n,m;
    struct Edge{
        int next,to;
        T w;
    };
    vector<Edge> edge;
    vector<int> head;
    int cnt;
    void add(int u,int v,T w){
        edge[++cnt].next = head[u];
        head[u] = cnt;
        edge[cnt].to = v;
        edge[cnt].w = w;
    }
    void init(int N,int M){
        n = N; m = M;
        head.clear();
        head.resize(n+1,0);
        edge.clear();
        edge.resize(M + 1);
        cnt = 0;
        return;
    }
};"\n"
```

Lca.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
const ll N = 2000000;
```

```
/*
O(1) LCA
先把 Graph 边练好,然后调用 Lca 类的 init 函数
然后就可以使用 lca(查祖先) 和 query(查距离) 了

注意边权不要爆 ll !
*/

class Graph{
public:
    int n;// 点的总数
    struct Edge{
        int next,to;
        ll w;
    }edge[N];
    int head[N],cnt;
    void add(int u,int v,ll w){
        edge[++cnt].next = head[u];
        head[u] = cnt;
        edge[cnt].to = v;
        edge[cnt].w = w;
    }
    void init(int N){
        n = N;
        for (int i = 1;i<n+1;i++){
            head[i] = 0;
        }
        cnt = 0;
        return;
    }
};

class Lca {
public:
    Graph* g;
    vector<int> depth, first, euler;
    vector<ll> dist;
    vector<vector<int>> st;
    int lg[2 * N];

    void init(Graph* graph, int root) {
        g = graph;
        euler.clear();
        depth.clear();
        dist.clear();
        dist.resize(g->n+1);
        first.assign(g->n + 1, -1);
        dfs(root, 0, 0, 0);
        build_st();
    }

    void dfs(int u, int fa, int d, ll sum) {
        first[u] = euler.size();
        euler.push_back(u);
```

```
        depth.push_back(d);
        dist[u] = sum;
        for (int i = g->head[u]; i; i = g->edge[i].next) {
            int v = g->edge[i].to;
            if (v == fa) continue;
            dfs(v, u, d + 1, sum + g->edge[i].w);
            euler.push_back(u);
            depth.push_back(d);
        }
    }

    void build_st() {
        int m = euler.size();
        int k = __lg(m) + 1;
        st.assign(k, vector<int>(m));
        for (int i = 0; i < m; ++i) st[0][i] = i;
        for (int i = 2; i < m + 5; ++i) lg[i] = lg[i >> 1] + 1;
        for (int j = 1; (1 << j) <= m; ++j)
            for (int i = 0; i + (1 << j) <= m; ++i) {
                int l = st[j - 1][i], r = st[j - 1][i + (1 << (j - 1))];
                st[j][i] = (depth[l] < depth[r] ? l : r);
            }
    }

    int lca(int u, int v) {
        int l = first[u], r = first[v];
        if (l > r) swap(l, r);
        int j = lg[r - l + 1];
        int a = st[j][l], b = st[j][r - (1 << j) + 1];
        return euler[depth[a] < depth[b] ? a : b];
    }

    ll query(int u,int v) {
        int LCA = lca(u,v);
        return dist[u] + dist[v] - 2 * dist[LCA];
    }
};

class Lca {
public:
    vector<vector<int>>* g;
    vector<int> depth, first, euler;
    vector<ll> dist;
    vector<vector<int>> st;
    int lg[2 * N];

    void init(vector<vector<int>>* graph, int root) {
        g = graph;
        euler.clear();
        depth.clear();
        dist.clear();
        dist.resize(g->size());
        first.assign(g->size(), -1);
        dfs(root, 0, 0, 0);
```

```
        build_st();
    }

    void dfs(int u, int fa, int d, ll sum) {
        first[u] = euler.size();
        euler.push_back(u);
        depth.push_back(d);
        dist[u] = sum;
        for (auto v : (*g)[u]) {
            if (v == fa) continue;
            dfs(v, u, d + 1, sum + 1); // 默认边权是 1
            euler.push_back(u);
            depth.push_back(d);
        }
    }

    void build_st() {
        int m = euler.size();
        int k = __lg(m) + 1;
        st.assign(k, vector<int>(m));
        for (int i = 0; i < m; ++i) st[0][i] = i;
        for (int i = 2; i < m + 5; ++i) lg[i] = lg[i >> 1] + 1;
        for (int j = 1; (1 << j) <= m; ++j)
            for (int i = 0; i + (1 << j) <= m; ++i) {
                int l = st[j - 1][i], r = st[j - 1][i + (1 << (j - 1))];
                st[j][i] = (depth[l] < depth[r] ? l : r);
            }
    }

    int lca(int u, int v) {
        int l = first[u], r = first[v];
        if (l > r) swap(l, r);
        int j = lg[r - l + 1];
        int a = st[j][l], b = st[j][r - (1 << j) + 1];
        return euler[depth[a] < depth[b] ? a : b];
    }

    int query(int u,int v) {
        int LCA = lca(u,v);
        return dist[u] + dist[v] - 2 * dist[LCA];
    }
} solver;

// dep 跟 dist 不要搞混了谢谢喵
struct Lca {
    int M = 20;
    vector<vector<int>> fa;
    vector<int> dep;
    void init(vector<vector<int>>& g, int rt) {
        int n = g.size() - 1;
        fa.clear();
        fa.resize(n + 1, vector<int>(M, 0));
        dep.clear();
        dep.resize(n + 1, 0);
```

```
        dep[0] = 0;
        auto dfs = [&](auto&&self, int u, int f) -> void {
            for (auto v : g[u]) {
                if (v == f) continue;
                fa[v][0] = u;
                dep[v] = dep[u] + 1;
                self(self, v, u);
            }
        };
        dfs(dfs, rt, 0);
        for (int i = 1;i<M;i++) {
            for (int j = 1;j<n+1;j++) {
                fa[j][i] = fa[fa[j][i-1]][i-1];
            }
        }
        return;
    }

    int lca(int x, int y) {
        if (dep[x] < dep[y]) swap(x, y);
        for (int i = M-1;i>=0;i--) {
            if (dep[fa[x][i]] >= dep[y]) {
                x = fa[x][i];
            }
        }
        if (x == y) return x;
        for (int i=M-1;i>=0;i--) {
            if (fa[x][i] != fa[y][i]) {
                x = fa[x][i], y = fa[y][i];
            }
        }
        return fa[x][0];
    }

    int query(int x, int y) {
        int z = lca(x, y);
        return dep[x] + dep[y] - 2 * dep[z];
    }
} solver;"\n"
```

LcaDoubling.cpp

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
const ll N = 2000000;
#define GRAPH_FOR(x) for (int i = graph.head[x];i;i = graph.edge[i].next)
class Graph{
public:
    struct Edge{
```

```
        int next,to,w;
    }edge[N];
    int head[N],cnt;
    Graph (int N = 0){
        for (int i = 0;i<N+1;i++){
            head[i] = 0;
        }
    }
    void init(int N){
        for (int i = 0;i<N+1;i++){
            head[i] = 0;
        }
    }
    void add(int u,int v,int w){
        edge[++cnt].next = head[u];
        head[u] = cnt;
        edge[cnt].to = v;
        edge[cnt].w = w;
    }
};
const ll M = 22;
class LcaDoubling{
public:
    Graph graph;
    int fa[N][M],n,dep[N];
    LcaDoubling(int N){
        n = N;
        for (int i = 0;i<n+1;i++){
            graph.head[i] = 0;
            dep[i] = 0;
        }
    }
    void dfs(int u){
        dep[u] = dep[fa[u][1]] + 1;
        fa[u][0] = u;
        GRAPH_FOR(u){
            int v = graph.edge[i].to;
            if (v == fa[u][1]){
                continue;
            }
            fa[v][1] = u;
            dfs(v);
        }
        return;
    }
    void init(){
        dfs(1);
        for (int i = 2;i<=M-1;i++){
            for (int j = 1;j<n+1;j++){
                fa[j][i] = fa[fa[j][i-1]][i-1];
            }
        }
    }
    int lca(int u,int v){
```

```
        if (dep[v] != dep[u]){
            if (dep[u] > dep[v]) swap(u,v);
            int tmp = dep[v] - dep[u];
            for (int i = 1;tmp;i++,tmp >>= 1){
                if (tmp & 1){
                    v = fa[v][i];
                }
            }
        }
        if (u == v){
            return u;
        }
        for (int i = M-1;i>=1;i--){
            if (fa[u][i] != fa[v][i]){
                u = fa[u][i],v = fa[v][i];
            }
        }
        return fa[u][1];
    }
};"\n"
```

MinCostMaxFlow.cpp

```
#include <bits/stdc++.h>
using namespace std;

template<typename T>
class MinCostMaxFlow {
public:
    struct Edge {
        int next, to;
        T w, c;
    };

    int n,cnt;
    vector<Edge> edge;
    vector<T> dis,flow;
    vector<int> pre,last,head;
    vector<bool> vis;


    MinCostMaxFlow(int n, int edgeCapacity = 4000000) : n(n) {
        head.assign(n + 1, 0);
        edge.resize(edgeCapacity + 1);
        cnt = 1;
        dis.assign(n + 1, 0);
        pre.assign(n + 1, -1);
        last.assign(n + 1, 0);
        flow.assign(n + 1, 0);
        vis.assign(n + 1, false);
    }
```

```
    void init(int n) {
        this->n = n;
        fill(head.begin(), head.end(), 0);
        cnt = 1;
    }

    void addEdge(int u, int v, T w,T c) {
        edge[++cnt] = { head[u], v, w, c };
        head[u] = cnt;
        edge[++cnt] = { head[v], u, 0, -c };
        head[v] = cnt;
    }

    bool spfa(int s, int t) {
        const int INF = 0x3f3f3f3f;
        fill(dis.begin(), dis.end(), INF);
        fill(flow.begin(), flow.end(), INF);
        fill(vis.begin(), vis.end(), false);
        fill(pre.begin(), pre.end(), -1);

        queue<int> q;
        q.push(s);
        vis[s] = true;
        dis[s] = 0;
        flow[s] = INF;

        while (!q.empty()) {
            int u = q.front();
            q.pop();
            vis[u] = false;
            for (int i = head[u]; i; i = edge[i].next) {
                int v = edge[i].to;
                if (edge[i].w > 0 && dis[v] > dis[u] + edge[i].c) {
                    dis[v] = dis[u] + edge[i].c;
                    pre[v] = u;
                    last[v] = i;
                    flow[v] = min(flow[u], edge[i].w);
                    if (!vis[v]) {
                        q.push(v);
                        vis[v] = true;
                    }
                }
            }
        }
        return pre[t] != -1;
    }

    pair<T, T> run(int s, int t) {
        T maxFlow = 0, minCost = 0;
        while (spfa(s, t)) {
            int f = flow[t];
            maxFlow += f;
            minCost += f * dis[t];
```

```
            for (int u = t; u != s; u = pre[u]) {
                int i = last[u];
                edge[i].w -= f;
                edge[i ^ 1].w += f;
            }
        }
        return { maxFlow, minCost };
    }
};"\n"
```

Prufer.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using i128 = __int128_t;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

/*
做模版题的时候发现这个 push_back 的常数真不是一般的大
数据量大的时候慎用这个 push_back 吧
*/
struct Prufer {
    // 注意 g 的下标从 1 开始
    // vector<int> getPrufer(vector<vector<int>>& g) {
    //  int n = g.size() - 1;
    //  vector<int> d(n + 1, 0);
    //  for (int i = 1;i<n+1;i++) {
    //      d[i] = g[i].size();
    //  }
    //  vector<int> fa(n + 1, 0);
    //  {
    //      vector<int> stk;
    //      stk.push_back(n);
    //      while (!stk.empty()) {
    //          int u = stk.back(); stk.pop_back();
    //          for (auto v : g[u]) {
    //              if (v == fa[u]) continue;
    //              fa[v] = u;
    //              stk.push_back(v);
    //          }
    //      }
    //  }
    //  vector<int> p;
    //  for (int i = 1;i<=n&&p.size()<n-2;i++) {
    //      if (d[i] == 1 && fa[i]) {
```

```
//          p.push_back(fa[i]);
//          d[fa[i]] --;
//          d[i] --;
//          int u = fa[i];
//          while (d[u] == 1 && u<i+1 && fa[u] && p.size()<n-2) {
//              p.push_back(fa[u]);
//              d[fa[u]] --;
//              d[u] --;
//              u = fa[u];
//          }
//      }
//  }
//  return p;
// }
vector<int> getPrufer(vector<int>& fa) {
    int n = fa.size() - 1;
    vector<int> d(n + 1, 0);
    for (int i = 1;i<n+1;i++) {
        if (fa[i]) {
            d[i] ++; d[fa[i]] ++;
        }
    }
    vector<int> p;
    for (int i = 1;i<=n&&p.size()<n-2;i++) {
        if (d[i] == 1 && fa[i]) {
            p.push_back(fa[i]);
            d[fa[i]] --;
            d[i] --;
            int u = fa[i];
            while (d[u] == 1 && u<i+1 && fa[u] && p.size()<n-2) {
                p.push_back(fa[u]);
                d[fa[u]] --;
                d[u] --;
                u = fa[u];
            }
        }
    }
    return p;
}

// 注意 p 的下标是从 0 开始
// 返回的 fa 数组是从 1 开始
vector<int> getTree(vector<int>& p) {
    int n = p.size() + 2;
    vector<int> d(n + 1, 1);
    for (auto v : p) {
        d[v] ++;
    }
    vector<int> fa(n + 1, 0);
    for (int i = 1,j=0;i<=n&&j<p.size();i++) {
        if (d[i] == 1) {
            fa[i] = p[j];
            d[i] --; d[p[j]] --;
            int u = p[j]; j ++;
```

```
                while (d[u] == 1&&j<p.size()&&u<i+1) {
                    fa[u] = p[j];
                    d[u] --; d[p[j]] --;
                    u = p[j]; j ++;
                }
            }
        }
        for (int i = 1;i<n;i++) {
            if (d[i] == 1) {
                fa[i] = n;
                break;
            }
        }
        return fa;
    }
} solver;"\n"
```

ShortestPath.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

const ll N = 2000000;
const ll MOD = 998244353;
const ll MAX = 1e18;

struct ShortestPath {
    int n;
    vector<vector<int>>& g;
    struct Node {
        ll val;
        int idx;
        bool operator<(const Node& A) const {

        };
    };
    priority_queue<Node> nodes;
    void init(vector<vector<int>>& G) {
        g = G;
        int n = g.size() + 1;
    }
    void cal() {
        while (!nodes.empty()) {
            Node u = nodes.top(); nodes.pop();
            for (auto v : g[u.idx]) {

            }
        }
    }
};
```

```
template<typename T>
class Graph{
public:
    int n,m;
    struct Edge{
        int next,to;
        T w;
    };
    vector<Edge> edge;
    vector<int> head;
    int cnt;
    void add(int u,int v,T w){
        edge[++cnt].next = head[u];
        head[u] = cnt;
        edge[cnt].to = v;
        edge[cnt].w = w;
    }
    void init(int N,int M){
        n = N; m = M;
        head.clear();
        head.resize(n+1,0);
        Edge.clear();
        Edge.resize(M + 1);
        cnt = 0;
        return;
    }
};

template<typename T>
class Dijkstra{
public:
    Graph<T>* g;
    struct Node{
        int dist,idx;
        bool operator<(const Node &a) const {
            return dist > a.dist;
        }
        Node(int d,int x) {dist = d,idx = x;}
    };

    ll dist[N];
    bool vis[N];

    void dijkstra(int pos){
        priority_queue<Node> pq;
        for (int i = 1;i<g->n+1;i++){
            dist[i] = MAX;
            vis[i] = 0;
        }
        dist[pos] = 0;
        pq.push(Node(0,pos));

        while (!pq.empty()){
```

```
            int u = pq.top().idx;
            pq.pop();
            if (vis[u]){
                continue;
            }
            vis[u] = 1;
            for (int i = g->head[u];i;i = g->edge[i].next){
                int v = g->edge[i].to;
                if (dist[v] > dist[u] + g->edge[i].w){
                    dist[v] = dist[u] + g->edge[i].w;
                    if (!vis[v]){
                        pq.push(Node(dist[v],v));
                    }
                }
            }
        }
        return;
    }
};

/*
check 用于检查负环同时构造新边权
solve 输入起点,然后 dist 数组存最短路
*/

template<typename T>
class Johnson {
public:
    using ll = long long;
    Graph<T>* g;
    vector<T> h;
    vector<ll> dist;
    vector<bool> vis;

    void init(Graph<T>* G_) {
        g = G_;
        h.assign(g->n + 1, 0);
        vis.assign(g->n + 1, false);
    }

    bool check() {
        int n = g->n;
        int m = g->m;
        g->edge[0] = typename Graph<T>::Edge(); // dummy
        for (int i = 1; i <= n; i++) {
            g->add(0, i, 0); // 从虚拟源点 0 连向每个点
        }

        queue<int> q;
        vector<int> cnt(n + 2), inq(n + 2, 0);
        h.assign(n + 2, numeric_limits<T>::max() / 2);
        h[0] = 0;
        q.push(0);
        inq[0] = 1;
```

```
        while (!q.empty()) {
            int u = q.front(); q.pop();
            inq[u] = 0;
            for (int i = g->head[u]; i; i = g->edge[i].next) {
                int v = g->edge[i].to;
                T w = g->edge[i].w;
                if (h[v] > h[u] + w) {
                    h[v] = h[u] + w;
                    if (!inq[v]) {
                        q.push(v);
                        inq[v] = 1;
                        if (++cnt[v] > n) return false;
                    }
                }
            }
        }

        // 重新权重变换
        for (int u = 1; u <= n; u++) {
            for (int i = g->head[u]; i; i = g->edge[i].next) {
                int v = g->edge[i].to;
                g->edge[i].w += h[u] - h[v];
            }
        }

        return true;
    }

    void solve(int s) {
        dist.assign(g->n + 1, numeric_limits<ll>::max());
        vis.assign(g->n + 1, false);
        priority_queue<pair<ll, int>, vector<pair<ll, int>>, greater<>> q;

        dist[s] = 0;
        q.emplace(0, s);
        while (!q.empty()) {
            auto [d, u] = q.top(); q.pop();
            if (vis[u]) continue;
            vis[u] = true;
            for (int i = g->head[u]; i; i = g->edge[i].next) {
                int v = g->edge[i].to;
                T w = g->edge[i].w;
                if (dist[v] > dist[u] + w) {
                    dist[v] = dist[u] + w;
                    q.emplace(dist[v], v);
                }
            }
        }

        for (int i = 1; i <= g->n; i++) {
            if (dist[i] < numeric_limits<ll>::max() / 2) {
                dist[i] = dist[i] - h[s] + h[i];
            }
```

```
        }
    }
};"\n"
```

SteinerTree.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
const ll N = 2000000;

template<typename T>
class Graph{
public:
    int n,m;
    struct Edge{
        int next,to;
        T w;
    };
    vector<Edge> edge;
    vector<int> head;
    int cnt;
    void add(int u,int v,T w){
        edge[++cnt].next = head[u];
        head[u] = cnt;
        edge[cnt].to = v;
        edge[cnt].w = w;
    }
    void init(int N,int M){
        n = N; m = M;
        head.clear();
        head.resize(n+1,0);
        Edge.clear();
        Edge.resize(M + 1);
        cnt = 0;
        return;
    }
};

template<typename T>
class SteinerTree {
public:
    Graph<T>* g;
    int n, k;
    vector<int> spe;
    vector<vector<T>> dp;
    vector<T> dist;
    vector<bool> vis;

    void init(Graph<T>* _g, const vector<int>& _spe) {
```

```
        g = _g; n = _g->n; k = _spe.size(); spe = _spe;
        int U = 1 << k;
        dp.assign(n+1, vector<T>(U, numeric_limits<T>::max()/4));
        dist.assign(n+1, numeric_limits<T>::max()/4);
        vis.assign(n+1, false);
        for (int i = 0; i < k; i++)
            dp[spe[i]][1<<i] = 0;
    }

    T solve() {
        int U = (1<<k) - 1;
        for (int S = 1; S <= U; S++) {
            for (int A = (S-1)&S; A; A = (A-1)&S)
                for (int i = 1; i <= n; i++)
                    dp[i][S] = min(dp[i][S], dp[i][A] + dp[i][S^A]);

            priority_queue<pair<T,int>, vector<pair<T,int>>, greater<>> pq;
            for (int i = 1; i <= n; i++) {
                dist[i] = dp[i][S];
                vis[i] = false;
                pq.emplace(dist[i], i);
            }
            while (!pq.empty()) {
                auto [d,u] = pq.top(); pq.pop();
                if (vis[u]) continue;
                vis[u] = true;
                for (int e = g->head[u]; e; e = g->edge[e].next) {
                    int v = g->edge[e].to; T w = g->edge[e].w;
                    if (dist[v] > d + w) {
                        dist[v] = d + w;
                        pq.emplace(dist[v], v);
                    }
                }
            }
            for (int i = 1; i <= n; i++)
                dp[i][S] = dist[i];
        }

        T ans = numeric_limits<T>::max()/4;
        for (int i = 1; i <= n; i++)
            ans = min(ans, dp[i][U]);
        return ans;
    }
};"\n"
```

Tarjan.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
```

```cpp
using pii = pair<int,int>;
using i128 = __int128_t;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

// inde 数组存的是合并后的新点编号
// 记得补充 info 的合并函数 !!!
// 其实 info 的合并完全可以写在外部，内部只实现一个缩点
// 默认建立正向边的啊
struct Tarjan {
    struct Info {
        Info() {

        }
    };
    int tot, idx;
    vector<int> dfn, low, s, inde;
    vector<vector<int>> e;
    vector<bool> vis;
    vector<Info> infos;
    void tarjan(int u, vector<vector<int>>& g) {
        dfn[u] = low[u] = ++idx;
        s.push_back(u);
        vis[u] = 1;
        for (auto v : g[u]) {
            if (!dfn[v]) {
                tarjan(v, g);
                low[u] = min(low[u], low[v]);
            } else if (vis[v]) {
                low[u] = min(low[u], dfn[v]);
            }
        }
        if (low[u] == dfn[u]) {
            tot ++;
            infos.push_back(Info());
            while (1) {
                inde[s.back()] = tot;
                // info merge

                if (s.back() == u) {
                    vis[s.back()] = 0;
                    s.pop_back();
                    break;
                }
                vis[s.back()] = 0;
                s.pop_back();
            }
        }
        return;
    }
    void init(vector<vector<int>>& g) {
        int n = g.size() - 1;
```

```
        idx = tot = 0;
        vector<int>().swap(dfn);
        dfn.resize(n + 1, 0);
        vector<int>().swap(low);
        low.resize(n + 1, 0);
        vector<int>().swap(inde);
        inde.resize(n + 1, 0);
        vector<bool>().swap(vis);
        vis.resize(n + 1, 0);
        vector<Info>().swap(infos);
        infos.resize(1, Info());
        vector<vector<int>>().swap(e);
        e.resize(n + 1);

        for (int i = 1;i<=n;i++) {
            if (!dfn[i]) {
                tarjan(i, g);
            }
        }

        for (int i = 1;i<=n;i++) {
            for (auto v : g[i]) {
                if (inde[v] == inde[i]) continue;
                e[inde[i]].push_back(inde[v]);
            }
        }
        return;
    }
} solver;"\n"
```

Tree.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;
class Tree{
public:
    vector<int> g[N];
    int n;
    int fa[N],dep[N],siz[N];
public:
    Tree(int N) {
        n = N;
    }
    void init(){
        for (int i = 1;i<n;i++){
            int u,v;
            cin >> u >> v;
            g[u].push_back(v);
            g[v].push_back(u);
```

```
        }
        return;
    }
    void dfs_pre(int u,int f){
        siz[u] = 1;
        fa[u] = f;
        dep[u] = dep[f] + 1;
        for (auto v:g[u]){
            if (v == f) continue;
            dfs_pre(v,u);
        }
        return;
    }
};

using pii = pair<int, int>;
pii getDiameter(vector<vector<int>>& g) {
    int s = 1, e;
    int mxDep = -1;
    auto dfs = [&](auto&&self, int u, int dep, int f) -> void {
        if (mxDep < dep) {
            mxDep = dep;
            e = u;
        }
        for (auto v : g[u]) {
            if (v == f) continue;
            self(self, v, dep + 1, u);
        }
        return;
    };
    dfs(dfs, 1, 0, 1);
    swap(s, e); mxDep = -1;
    dfs(dfs, s, 0, s);
    return {s, e};
}"\n"
```

TreeChainSeg.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;

class Graph{
public:
    struct Edge{
        int next,to,w;
    }edge[N];
    int head[N],cnt;
    Graph (int N = 0){
        for (int i = 0;i<N+1;i++){
```

```
            head[i] = 0;
        }
    }
    void init(int N){
        for (int i = 0;i<N+1;i++){
            head[i] = 0;
        }
    }
    void add(int u,int v,int w){
        edge[++cnt].next = head[u];
        head[u] = cnt;
        edge[cnt].to = v;
        edge[cnt].w = w;
    }
};

class SegTree{
public:
    Graph* graph;
    struct Node{
    }nodes[N*4];
    #define ls (rt << 1)
    #define rs (rt << 1 | 1)
    Node merge(Node L,Node R){
        Node M;
        return M;
    }
    void build(int rt,int l,int r){
        if (l == r){
            return;
        }
        int mid = l + r >> 1;
        build(ls,l,mid),build(rs,mid+1,r);
        nodes[rt] = merge(nodes[ls],nodes[rs]);
    }
    void pd(int rt,int l,int r){
    }
    void update(int rt,int l,int r,int ql,int qr,int v){
        if (ql <= l && r <= qr){
            return;
        }
        int mid = l+r>>1;
        pd(rt,l,r);
        if (ql <= mid){
            update(ls,l,mid,ql,qr,v);
        }
        if (qr >= mid + 1){
            update(rs,mid+1,r,ql,qr,v);
        }
        nodes[rt] = merge(nodes[ls],nodes[rs]);
        return;
    }
    Node query(int rt,int l,int r,int ql,int qr){
        if (ql <= l && r <= qr){
```

```
            return nodes[rt];
        }
        int mid = l+r>>1;
        pd(rt,l,r);
        if (ql > mid){
            return query(rs,mid+1,r,ql,qr);
        }else if (qr < mid + 1){
            return query(ls,l,mid,ql,qr);
        }else{
            return merge(query(ls,l,mid,ql,qr),query(rs,mid+1,r,ql,qr));
        }
    }
};

class TreeChainSeg{
public:
    Graph* graph;
    SegTree segTree;
    int fa[N],top[N],siz[N],son[N],dep[N],dfn[N],id[N],cnt,n,cnt2;
    TreeChainSeg(int N){
        graph->init(N);
        n = N;
        cnt = 0;
        cnt2 = n+1;
        for (int i = 0;i<=N;i++){
            fa[i] = top[i] = siz[i] = son[i] = dep[i] = dfn[i] = id[i] = 0;
        }
    }
    void dfs1(int u){
        siz[u] = 1;
        for (int i = graph->head[u];i;i=graph->edge[i].next){
            int v = graph->edge[i].to;
            if (fa[u] == v) continue;
            dep[v] = dep[u] + 1;
            fa[v] = u;
            dfs1(v);
            siz[u] += siz[v];
            if (!son[u] || siz[son[u]] < siz[v]){
                son[u] = v;
            }
        }
    }
    void dfs2(int u,int tp){
        dfn[cnt] = u;
        id[u] = ++cnt;
        top[u] = tp;
        if (son[u]){
            dfs2(son[u],tp);
        }
        for (int i = graph->head[u];i;i=graph->edge[i].next){
            int v = graph->edge[i].to;
            if (v == fa[u] || v == son[u]) continue;
            dfs2(v,v);
        }
```

```
    }
    void init(Graph* g){
        graph = g;
        dfs1(1),dfs2(1,1);
        segTree.graph = graph;
        segTree.build(1,1,n);
    }
    int lca(int u,int v){
        while (top[u] != top[v]){
            if (dep[top[u]] > dep[top[v]]){
                u = fa[top[u]];
            }else{
                v = fa[top[v]];
            }
        }
        if (dep[u] > dep[v]){
            return v;
        }else{
            return u;
        }
    }
};"\n"
```

ArrayInversion.cpp

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

class ArrayInversion {
    ll mergeSort(vector<ll>& a, vector<ll>& temp, int l, int r) {
        if (l >= r) return 0;
        int mid = (l + r) / 2;
        ll inv_count = mergeSort(a, temp, l, mid) + mergeSort(a, temp, mid + 1,
r);
        int i = l, j = mid + 1, k = l;
        while (i <= mid && j <= r) {
            if (a[i] > a[j]) {
                temp[k++] = a[j++];
                inv_count += mid - i + 1;
            } else {
                temp[k++] = a[i++];
            }
        }
        while (i <= mid) temp[k++] = a[i++];
        while (j <= r) temp[k++] = a[j++];
        for (int i = l; i <= r; ++i) a[i] = temp[i];
        return inv_count;
    }

public:
```

```
    ll solve(vector<ll> a) {
        int n = a.size();
        vector<ll> temp(n);
        return mergeSort(a, temp, 0, n - 1);
    }
};
"\n"
```

AutoMod.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using i128 = __int128_t;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

template<ll M>
struct AutoMod {
    ll v;
    AutoMod(ll x=0){ v = x%M; if(v<0) v+=M; }
    static ll qpow(ll a,ll b){
        ll r=1, m=M;
        if(b<0) return qpow(qpow(a,-b),M-2);
        while(b){ if(b&1) r=r*a%m; a=a*a%m; b>>=1; }
        return r;
    }
    AutoMod pow(ll k)const{ return qpow(v,k); }
    AutoMod inv()const{ return qpow(v,M-2); }
    AutoMod operator+(AutoMod o) const{ return AutoMod(v+o.v); }
    AutoMod operator-(AutoMod o) const{ return AutoMod(v-o.v); }
    AutoMod operator*(AutoMod o) const{ return AutoMod(v*o.v); }
    AutoMod operator/(AutoMod o) const{ return *this * o.inv(); }
    bool operator==(AutoMod o) const{ return v==o.v; }
};
"\n"
```

BGSG.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using i128 = __int128_t;
```

```cpp
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

// 调用 cal(a, n, p) 输出 (a^x = n) % p
// 复杂度 sqrt(p) * log
struct ExBSGS {
    static ll modpow(ll a, ll e, ll mod){
        ll r = 1;
        a %= mod;
        while(e){
            if(e & 1) r = (i128)r * a % mod;
            a = (i128)a * a % mod;
            e >>= 1;
        }
        return r;
    }
    static ll exgcd(ll a, ll b, ll &x, ll &y){
        if(b == 0){ x = 1; y = 0; return a; }
        ll x1, y1;
        ll g = exgcd(b, a % b, x1, y1);
        x = y1;
        y = x1 - a / b * y1;
        return g;
    }
    static ll invmod(ll a, ll mod){
        ll x, y;
        ll g = exgcd(a, mod, x, y);
        if(g != 1) return -1;
        x %= mod;
        if(x < 0) x += mod;
        return x;
    }
    static ll bsgs(ll a, ll b, ll mod){
        a %= mod; b %= mod;
        if(mod == 1) return 0;
        ll m = (ll)ceil(sqrt((double)mod));
        unordered_map<ll, ll> mp;
        mp.reserve(m * 2);
        ll aj = 1;
        for(ll j = 0; j < m; ++j){
            if(mp.find(aj) == mp.end()) mp[aj] = j;
            aj = (i128)aj * a % mod;
        }
        ll factor = modpow(a, m, mod);
        ll invfactor = invmod(factor, mod);
        if(invfactor == -1) return -1;
        ll cur = b % mod;
        for(ll i = 0; i <= m; ++i){
            auto it = mp.find(cur);
            if(it != mp.end()){
                return i * m + it->second;
            }
```

```
            cur = (i128)cur * invfactor % mod;
        }
        return -1;
    }
    static ll cal(ll a, ll n, ll p){
        if(p == 1) return 0;
        a %= p; n %= p;
        if(n == 1) return 0;
        ll cnt = 0;
        ll t = 1;
        ll g;
        while((g = std::gcd(a, p)) > 1){
            if(n == t) return cnt;
            if(n % g != 0) return -1;
            p /= g;
            n /= g;
            t = (i128)t * (a / g) % p;
            ++cnt;
        }
        ll invt = invmod(t, p);
        if(invt == -1) return -1;
        ll rhs = (i128)n * invt % p;
        ll res = bsgs(a, rhs, p);
        if(res == -1) return -1;
        return res + cnt;
    }
};
"\n"
```

CombinationNumber.cpp

```
#include <bits/stdc++.h>
typedef long long ll;
const ll MOD = 1000000007;
// 1e7 的数组慎开啊
const ll MAXN = 10001000;
using namespace std;
class CombinationNumber{
public:
    ll fact[MAXN];
    ll inv[MAXN];
    ll quick(ll base, ll k) {
        ll res = 1;
        while (k) {
            if (k & 1) {
                res = res * base % MOD;
            }
            base = base * base % MOD;
            k >>= 1;
        }
        return res;
```

```
    }

    void precompute() {
        ll Z = min(MAXN, MOD);
        fact[0] = 1;
        for (int i = 1; i < Z; i++) {
            fact[i] = fact[i - 1] * i % MOD;
        }
        inv[Z - 1] = quick(fact[Z - 1], MOD - 2);
        for (int i = Z - 2; i >= 0; i--) {
            inv[i] = inv[i + 1] * (i + 1) % MOD;
        }
    }

    ll C(ll n, ll m) {
        if (m > n || m < 0) {
            return 0;
        }
        return fact[n] * inv[m] % MOD * inv[n - m] % MOD;
    }

    void init(){
        precompute();
    }
} solver2;"\n"
```

Crt.cpp

```
/*
功能简短说明：

1. 输入：两个 `(r,m)` 分别表示 `x ≡ r (mod m)`；建议 `m>0` 且 `0≤r<m`。
2. 支持 **模不互质** 的情况（会处理公约数）。
3. 输出：若有解返回 `(r,M)`，表示所有解为 `x = r + k*M`（`k∈Z`），且 `0 ≤ r < M`；若无解返回 `(-1,-1)`。
4. 合并规则：如果方程组相容（即 `r2-r1` 能被 `g = gcd(m1,m2)` 整除），则合并得到模 `M = m1 * (m2 / g)`，并计算最小非负解 `r`。
5. 算法要点：用扩展欧几里得求 `g` 和系数 `x,y`，检查 `(r2-r1)%g==0`，计算位移 `t = (r2-r1)/g * x (mod m2/g)`，再算出 `r = (r1 + m1*t) mod M`。实现中用 `__int128` 防止乘法溢出。
   示例：`(2,3)` 与 `(3,5)` 合并得 `(8,15)`，因为解集是 `x ≡ 8 (mod 15)`。
*/
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
using i128 = __int128;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
ll exgcd(ll a,ll b,ll &x,ll &y){
    if(b==0){x=1;y=0;return a;}
    ll x1,y1; ll g=exgcd(b,a%b,x1,y1);
```

```
    x=y1; y=x1-(a/b)*y1; return g;
}
pll crt(pll A,pll B){
    ll r1=A.first,m1=A.second,r2=B.first,m2=B.second;
    if(m1<=0||m2<=0) return {-1,-1};
    r1%=m1; if(r1<0) r1+=m1;
    r2%=m2; if(r2<0) r2+=m2;
    ll x,y; ll g=exgcd(m1,m2,x,y);
    ll d=r2-r1; if(d%g!=0) return {-1,-1};
    i128 t=(i128)(d/g)*(i128)x;
    i128 mod2=m2/g; t%=mod2; if(t<0) t+=mod2;
    i128 M=(i128)m1*mod2;
    i128 res=((i128)r1 + (i128)m1 * t) % M; if(res<0) res+=M;
    return {(ll)res,(ll)M};
}
pll crt_many(const vector<pll>& v){
    if(v.empty()) return {0,1};
    pll cur = v[0];
    for(size_t i=1;i<v.size();++i){
        cur = crt(cur,v[i]);
        if(cur.first==-1) return cur;
    }
    return cur;
}
ll inv_mod(ll a, ll b){
    ll x, y;
    ll g = exgcd(a, b, x, y);
    if(g != 1) return -1;
    x %= b;
    if(x < 0) x += b;
    return x;
}
int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n; if(!(cin>>n)) return 0;
    vector<pll> v; v.reserve(n);
    for(int i=0;i<n;++i){ ll r,m; cin>>r>>m; v.emplace_back(r,m); }
    auto ans = crt_many(v);
    cout<<ans.first<<" "<<ans.second<<"\n";
    return 0;
}
"\n"
```

ExGcd.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

/*
```

```
使用说明:
首先调用 init 函数初始化 ax + by = c 的参数,返回 0 表示无解.
cal 函数计算特解,通解为 X1 + k * dx , Y1 + k * dy
*/
class ExGcd{
public:
    int a,b,dx,dy,c,gcd_ab;
    int X0,Y0,X1,Y1;
    bool init(int A,int B,int C = 0){
        a = A,b = B,c = C;
        gcd_ab = __gcd(a,b);
        if (c % gcd_ab != 0){
            return 0;
        }
        return 1;
    }
    void exgcd(int a,int b){
        if (b == 0){
            X0 = 1;
            Y0 = 0;
        }else{
            exgcd(b,a % b);
            X1 = X0,Y1 = Y0;
            X0 = Y1;
            Y0 = X1 - a / b * Y1;
        }
        return;
    }
    void cal(){
        exgcd(a,b);
        X1 = X0 * c / gcd_ab;
        Y1 = Y0 * c / gcd_ab;
        dx = b / gcd_ab;
        dy = -a / gcd_ab;
        return;
    }
};"\n"
```

Hash.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2000000;


// 双模数 hash, 记得先 init
struct Hash {
    int x,y,MOD1 = 1000000007,MOD2 = 1000000009;
    Hash(){x = y = 0;}
    Hash(int _x,int _y) { x = _x; y = _y; }
```

```
    Hash operator + (const Hash &a) const {
        return Hash((x + a.x) % MOD1,(y + a.y) % MOD2);
    }
    Hash operator - (const Hash &a) const {
        return Hash((x - a.x + MOD1) % MOD1,(y - a.y + MOD2) % MOD2);
    }
    Hash operator * (const Hash &a) const {
        return Hash(1ll * x * a.x % MOD1,1ll * y * a.y % MOD2);
    }
    Hash operator * (const ll &a) const {
        return Hash(1ll * x * a % MOD1,1ll * y * a % MOD2);
    }
    bool operator == (const Hash &a) const {
        return (x == a.x && y == a.y);
    }
    bool operator<(const Hash& a) const {
        if (x != a.x) {
            return x < a.x;
        }
        return y < a.y;
    }
    bool operator>(const Hash& a) const {
        if (x != a.x) {
            return x > a.x;
        }
        return y > a.y;
    }
}base(131,13331),hs[N],bs[N];

void hash_init(int n){
    bs[0] = Hash(1,1);
    for(int i = 1;i <= n;i ++) {
        bs[i] = bs[i-1] * base;
    }
}"\n"
```

Int128.cpp

```
#include <bits/stdc++.h>
using namespace std;

typedef __int128 int128;

// 快速读入 __int128
int128 readInt128() {
    int128 n = 0;
    bool negative = false;
    char c = getchar();
    while (c < '0' || c > '9') {
        if (c == '-') negative = true;
        c = getchar();
```

```
    }
    while (c >= '0' && c <= '9') {
        n = n * 10 + (c - '0');
        c = getchar();
    }
    return negative ? -n : n;
}

// 快速输出 __int128
void printInt128(int128 n) {
    if (n < 0) {
        putchar('-');
        n = -n;
    }
    if (n == 0) {
        putchar('0');
        return;
    }
    char buffer[100];
    int bufferIndex = 0;
    while (n > 0) {
        buffer[bufferIndex++] = (n % 10) + '0';
        n /= 10;
    }
    while (bufferIndex > 0) {
        putchar(buffer[--bufferIndex]);
    }
}
"\n"
```

LagrangeInterpolation.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
const ll N = 4000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

namespace LagInt {
    ll qpow(ll base,ll k,ll mod) {
        ll res = 1;
        if (k < 0) {
            return qpow(qpow(base,-k,mod),mod-2,mod);
        }
        while (k) {
            if (k & 1) {
                res *= base; res %= mod;
```

```
        }
        k >>= 1;
        base *= base; base %= mod;
    }
    return res;
}

ll fac[N],facInv[N];

void init(int n) {
    fac[0] = 1; facInv[0] = 1;
    for (int i = 1;i<=n;i++) {
        fac[i] = fac[i - 1] * i % MOD;
    }

    vector<ll> pre(n + 1,1),suc(n + 1,1);
    for (int i = 1;i<=n;i++) {
        pre[i] = pre[i - 1] * fac[i] % MOD;
    }
    suc[n] = fac[n];
    for (int i = n - 1;i>=0;i--) {
        suc[i] = suc[i + 1] * fac[i] % MOD;
    }

    ll S = qpow(pre[n],MOD - 2,MOD);

    for (int i = 1;i<=n;i++) {
        facInv[i] = S * pre[i - 1] % MOD * suc[i + 1] % MOD;
    }
    return;
}

// 当 x[i] = i 时,用 A 模式,复杂度 nlogn
// 记得先调用预处理
ll LagrangeInterpolationA(ll x,vector<ll>& a) {
    x %= MOD;
    if (x < 0) return 0;
    if (a.size() > x) {
        return a[x];
    }
    ll t = 1;
    for (int i = 1;i<=a.size() - 1;i++) {
        t *= (x - i); t %= MOD;
    }

    ll ans = 0;
    int n = int(a.size()) - 1;

    for (int i = 1;i<a.size();i++) {
        ll y = a[i];
        ll res = t;
        res *= y; res %= MOD;
        res *= qpow((x - i) % MOD,MOD-2,MOD); res %= MOD;
        res *= facInv[i - 1]; res %= MOD;
```

```
            res *= facInv[n - i]; res %= MOD;
            ll sign = ((n - i) & 1) ? (MOD - 1) : 1;
            res = res * sign % MOD;
            ans += res; ans %= MOD;
        }
        return ans;
    }
};

void solve() {
    ll n,k;
    cin >> n >> k;
    LagInt::init(2e6);

    vector<ll> a(k + 3,0);
    for (int i = 1;i<k+3;i++) {
        a[i] = a[i - 1] + LagInt::qpow(i,k,MOD);
        a[i] %= MOD;
    }

    cout << LagInt::LagrangeInterpolationA(n,a) << endl;
    return;
}

signed main() {
#if DEBUG
    freopen("input.txt", "r", stdin);
    auto start_time = chrono::steady_clock::now();
#else
    ios::sync_with_stdio(false);
#endif
    cin.tie(nullptr);

    int t = 1;
    // cin >> t;

    while (t--) {
        solve();
    }

#if DEBUG
    auto end_time = chrono::steady_clock::now();
    auto diff = chrono::duration_cast<chrono::milliseconds>(end_time -
start_time);
    cerr << "Time: " << diff.count() << " ms" << endl;

#endif

    return 0;
}
"\n"
```

LinearBasis.cpp

```cpp
#include <bits/stdc++.h>
using ll = long long;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;
using namespace std;

struct LinearBasis {
    ll basis[63];
    bool flag0; // 是否可以表示 0

    void insert(ll x) {
        if (!x) return;
        for (int i = 60;i>=0;i--) {
            if ((1ll<<i)&x) {
                if (basis[i]) {
                    x ^= basis[i];
                } else {
                    basis[i] = x;
                    break;
                }
            }
        }
        if (x) flag0 = 1;
        return;
    }

    void preprocess() {
        for (int i = 60;i>=0;i--) {
            if (!basis[i]) continue;
            for (int j = i-1;j>=0;j--) {
                if (basis[j] && ((1ll<<j)&basis[i])) {
                    basis[i] ^= basis[j];
                }
            }
        }
        return;
    }

    void init() {
        flag0 = 0;
        memset(basis,0,sizeof basis);
        return;
    }

    ll get_mn() {
        for (int i = 0;i<=60;i++) {
            if (basis[i]) {
                return basis[i];
            }
```

```
        }
    }

    ll get_mx() {
        ll res = 0;
        for (int i = 60;i>=0;i--) {
            if (basis[i] && !((1ll<<i)&res)){
                res ^= basis[i];
            }
        }
        return res;
    }
} ;"\n"
```

Lucas.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using i128 = __int128_t;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

ll qpow(ll base,ll k,ll mod) {
    ll res = 1;
    if (k < 0) {
        return qpow(qpow(base,-k,mod),mod-2,mod);
    }
    while (k) {
        if (k & 1) {
            res *= base; res %= mod;
        }
        k >>= 1;
        base *= base; base %= mod;
    }
    return res;
}

// cal 函数是用于调用的接口, C 函数是内部使用的!!!
struct Lucas {
    int p;
    vector<int> fact;
    void init(int P_) {
        p = P_;
        fact.resize(p);
        fact[0] = 1;
        for (int i = 1;i<p;i++) {
```

```
            fact[i] = 1ll * fact[i - 1] * i % p;
        }
    }
    int C(int n, int m) {
        if (m > n) return 0;
        return 1ll * fact[n] * qpow(fact[n-m], p-2, p) % p * qpow(fact[m], p-2, p)
% p;
    }
    int cal(int n, int m) {
        if (m > n) return 0;
        if (n < p && m < p) {
            return C(n, m);
        }
        return 1ll * cal(n / p, m / p) * C(n % p, m % p) % p;
    }
};

// 这个 C 真的是接口了
struct ExLucas {
    ll mod = 1;
    vector<ll> p, e, pk, prod;
    void init(ll M) {
        mod = M;
        ll x = M;
        for (ll i = 2; i * i <= x; i++) if (x % i == 0) {
            ll cnt = 0, pw = 1;
            while (x % i == 0) { x /= i; cnt++; pw *= i; }
            p.push_back(i); e.push_back(cnt); pk.push_back(pw);
        }
        if (x > 1) { p.push_back(x); e.push_back(1); pk.push_back(x); }
        prod.resize(pk.size());
        for (size_t idx = 0; idx < pk.size(); ++idx) {
            ll P = p[idx], PK = pk[idx], pr = 1;
            for (ll i = 1; i <= PK; i++) if (i % P) pr = (i128)pr * i % PK;
            prod[idx] = pr;
        }
    }
    ll modpow(ll a, ll b, ll m) {
        ll r = 1 % m;
        a %= m;
        while (b) {
            if (b & 1) r = (i128)r * a % m;
            a = (i128)a * a % m;
            b >>= 1;
        }
        return r;
    }
    ll exgcd(ll a, ll b, ll &x, ll &y) {
        if (b == 0) { x = 1; y = 0; return a; }
        ll x1, y1; ll g = exgcd(b, a % b, x1, y1);
        x = y1; y = x1 - (a / b) * y1; return g;
    }
    ll invmod(ll a, ll m) { ll x, y; ll g = exgcd((a % m + m) % m, m, x, y);
return g == 1 ? (x % m + m) % m : -1; }
```

```
    ll vp(ll n, ll P) { ll cnt = 0; while (n) { n /= P; cnt += n; } return cnt; }
    ll fac_mod(ll n, ll idx) {
        ll P = p[idx], PK = pk[idx], PR = prod[idx];
        if (n == 0) return 1 % PK;
        ll res = modpow(PR, n / PK, PK);
        for (ll i = 1; i <= n % PK; i++) if (i % P) res = (i128)res * i % PK;
        return (i128)res * fac_mod(n / P, idx) % PK;
    }
    ll C_mod_pk(ll n, ll m, ll idx) {
        if (m < 0 || m > n) return 0;
        ll P = p[idx], PK = pk[idx], K = e[idx];
        ll cnt = vp(n, P) - vp(m, P) - vp(n - m, P);
        if (cnt >= K) return 0;
        ll a = fac_mod(n, idx);
        ll b = fac_mod(m, idx) * fac_mod(n - m, idx) % PK;
        ll ib = invmod(b, PK);
        ll res = (i128)a * ib % PK;
        res = (i128)res * modpow(P, cnt, PK) % PK;
        return res;
    }
    ll C(ll n, ll m) {
        if (m < 0 || m > n) return 0;
        int sz = pk.size();
        if (sz == 0) return 1 % mod;
        vector<ll> r(sz);
        for (int i = 0; i < sz; i++) r[i] = C_mod_pk(n, m, i);
        ll M = mod, x = 0;
        for (int i = 0; i < sz; i++) {
            ll mi = pk[i], ai = r[i];
            ll Mi = M / mi;
            ll ti = invmod(Mi % mi, mi);
            x = (x + (i128)ai * Mi % M * ti) % M;
        }
        return (x % M + M) % M;
    }
};"\n"
```

Mat.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

template <typename T>
struct Mat
{
    int n, m;
    T **a;
    Mat(int _n = 0, int _m = 0) : n(_n), m(_m)
    {
        a = new T *[n];
```

```
        for (int i = 0; i < n; i++)
            a[i] = new T[m], memset(a[i], 0, sizeof(T) * m);
    }
    Mat(const Mat &B)
    {
        n = B.n, m = B.m;
        a = new T *[n];
        for (int i = 0; i < n; i++)
            a[i] = new T[m], memcpy(a[i], B.a[i], sizeof(T) * m);
    }
    ~Mat() { delete[] a; }
    Mat &operator=(const Mat &B)
    {
        delete[] a;
        n = B.n, m = B.m;
        a = new T *[n];
        for (int i = 0; i < n; i++)
            a[i] = new T[m], memcpy(a[i], B.a[i], sizeof(T) * m);
        return *this;
    }
    Mat operator+(const Mat &B) const
    {
        assert(n == B.n && m == B.m);
        Mat ret(n, m);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                ret.a[i][j] = (a[i][j] + B.a[i][j]) % mod;
        return ret;
    }
    Mat &operator+=(const Mat &B) { return *this = *this + B; }
    Mat operator*(const Mat &B) const
    {
        Mat ret(n, B.m);
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < B.m; ret.a[i][j++] %= mod)
                for (int k = 0; k < m; ++k)
                    ret.a[i][j] += a[i][k] * B.a[k][j] % mod;
        return ret;
    }
    Mat &operator*=(const Mat &B) { return *this = *this * B; }
};
Mat<ll> qpow(Mat<ll> A, ll b)
{
    Mat<ll> ret(A);
    for (--b; b; b >>= 1, A *= A)
        if (b & 1)
            ret *= A;
    return ret;
}"\n"
```

NumberTheoryBlock.cpp

```
#include <bits/stdc++.h>
using namespace std;

using ll = long long;

class NumberTheoryBlock {
public:
    void down(ll x,vector<pair<ll,ll>>& block) {
        block.clear();
        ll l = 1,r = 0;
        while (l <= x) {
            r = x / (x / l);
            if (r > x) r = x;
            block.emplace_back(l,r);
            l = r + 1;
        }
        return;
    }

    void up(ll x,vector<pair<ll,ll>>& block) {
        block.clear();
        ll l = 1,r = 0;
        while (l <= x) {
            if (l == x) {
                block.emplace_back(l,l);
                break;
            }
            r = (x - 1) / ((x - 1) / l);
            block.emplace_back(l,r);
            l = r + 1;
        }
        return;
    }
};"\n"
```

Poly.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using i128 = __int128_t;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

namespace polystd {
#ifdef LOCAL
```

```
#define debug(...) fprintf(stderr, ##__VA_ARGS__)
#else
#define endl "\n"
#define debug(...) void(0)
#endif
typedef long long LL;

template <unsigned umod>
struct modint {
  static constexpr int mod = umod;
  unsigned v;
  modint() : v(0) {}
  template <class T, enable_if_t<is_integral<T>::value>* = nullptr>
  modint(T x) {
    x %= mod;
    if (x < 0) x += mod;
    v = x;
  }
  modint(const string& str) {
    v = 0;
    size_t i = 0;
    if (str.front() == '-') i += 1;
    while (i < str.size()) {
      assert(isdigit(str[i]));
      v = (v * 10ull % umod + str[i] - '0') % umod;
      i += 1;
    }
    if (str.front() == '-' && v) v = umod - v;
  }
  modint operator+() const { return *this; }
  modint operator-() const { return modint() - *this; }
  friend int raw(const modint& self) { return self.v; }
  friend istream& operator>>(istream& is, modint& self) {
    string str;
    is >> str;
    self = str;
    return is;
  }
  friend ostream& operator<<(ostream& os, const modint& self) {
    return os << raw(self);
  }
  modint& operator+=(const modint& rhs) {
    v += rhs.v;
    if (v >= umod) v -= umod;
    return *this;
  }
  modint& operator-=(const modint& rhs) {
    v -= rhs.v;
    if (v >= umod) v += umod;
    return *this;
  }
  modint& operator*=(const modint& rhs) {
    v = static_cast<unsigned>(1ull * v * rhs.v % umod);
    return *this;
```

```
  }
  modint& operator/=(const modint& rhs) {
    static constexpr size_t ilim = 1 << 20;
    static modint inv[ilim + 10];
    static int sz = 0;
    assert(rhs.v);
    if (rhs.v > ilim) return *this *= qpow(rhs, mod - 2);
    if (!sz) inv[1] = sz = 1;
    while (sz < (int)rhs.v) {
      for (int i = sz + 1; i <= sz << 1; i++) inv[i] = -mod / i * inv[mod % i];
      sz <<= 1;
    }
    return *this *= inv[rhs.v];
  }
  template <class T>
  friend modint qpow(modint a, T b) {
    modint r = 1;
    for (; b; b >>= 1, a *= a)
      if (b & 1) r *= a;
    return r;
  }
  friend modint operator+(modint lhs, const modint& rhs) { return lhs += rhs; }
  friend modint operator-(modint lhs, const modint& rhs) { return lhs -= rhs; }
  friend modint operator*(modint lhs, const modint& rhs) { return lhs *= rhs; }
  friend modint operator/(modint lhs, const modint& rhs) { return lhs /= rhs; }
  friend bool operator==(const modint& lhs, const modint& rhs) {
    return lhs.v == rhs.v;
  }
  friend bool operator!=(const modint& lhs, const modint& rhs) {
    return lhs.v != rhs.v;
  }
};

typedef modint<998244353> mint;
int glim(const int& x) { return 1 << (32 - __builtin_clz(x - 1)); }
int bitctz(const int& x) { return __builtin_ctz(x); }
struct poly : vector<mint> {
  poly() {}
  explicit poly(int n) : vector<mint>(n) {}
  poly(const vector<mint>& vec) : vector<mint>(vec) {}
  poly(initializer_list<mint> il) : vector<mint>(il) {}
  mint operator()(const mint& x) const;
  poly& cut(int lim);
  void ntt(int op);
};
void print(const poly& a) {
  for (size_t i = 0; i < a.size(); i++) debug("%d, ", raw(a[i]));
  debug("\n");
}
istream& operator>>(istream& is, poly& a) {
  for (auto& x : a) is >> x;
  return is;
}
ostream& operator<<(ostream& os, const poly& a) {
```

```
  bool flag = false;
  for (auto& x : a) {
    if (flag)
      os << " ";
    else
      flag = true;
    os << x;
  }
  return os;
}
mint poly::operator()(const mint& x) const {
  const auto& a = *this;
  mint res = 0;
  for (int i = (int)a.size() - 1; i >= 0; i--) {
    res = res * x + a[i];
  }
  return res;
}
poly& poly::cut(int lim) {
  resize(lim);
  return *this;
}
void poly::ntt(int op) {
  static bool wns_flag = false;
  static vector<mint> wns;
  if (!wns_flag) {
    wns_flag = true;
    for (int j = 1; j <= 23; j++) {
      wns.push_back(qpow(mint(3), raw(mint(-1)) >> j));
    }
  }
  vector<mint>& a = *this;
  int n = a.size();
  for (int i = 1, r = 0; i < n; i++) {
    r ^= n - (1 << (bitctz(n) - bitctz(i) - 1));
    if (i < r) std::swap(a[i], a[r]);
  }
  vector<mint> w(n);
  for (int k = 1, len = 2; len <= n; k <<= 1, len <<= 1) {
    mint wn = wns[bitctz(k)];
    for (int i = raw(w[0] = 1); i < k; i++) w[i] = w[i - 1] * wn;
    for (int i = 0; i < n; i += len) {
      for (int j = 0; j < k; j++) {
        mint x = a[i + j], y = a[i + j + k] * w[j];
        a[i + j] = x + y, a[i + j + k] = x - y;
      }
    }
  }
  if (op == -1) {
    mint iz = mint(1) / n;
    for (int i = 0; i < n; i++) a[i] *= iz;
    reverse(a.begin() + 1, a.end());
  }
}
```

```
poly concalc(int n, vector<poly> vec,
             const function<mint(vector<mint>)>& func) {
  int lim = glim(n);
  int m = vec.size();
  for (auto& f : vec) f.resize(lim), f.ntt(1);
  vector<mint> tmp(m);
  poly ret(lim);
  for (int i = 0; i < lim; i++) {
    for (int j = 0; j < m; j++) tmp[j] = vec[j][i];
    ret[i] = func(tmp);
  }
  ret.ntt(-1);
  return ret;
}
poly getInv(const poly& a, int lim) {
  poly b{1 / a[0]};
  for (int len = 2; len <= glim(lim); len <<= 1) {
    poly c = vector<mint>(a.begin(), a.begin() + min(len, (int)a.size()));
    b = concalc(len << 1, {b, c}, [](vector<mint> vec) {
          return vec[0] * (2 - vec[0] * vec[1]);
        }).cut(len);
  }
  return b.cut(lim);
}
poly operator+=(poly& a, const poly& b) {
  if (a.size() < b.size()) a.resize(b.size());
  for (size_t i = 0; i < b.size(); i++) a[i] += b[i];
  return a;
}
poly operator-=(poly& a, const poly& b) {
  if (a.size() < b.size()) a.resize(b.size());
  for (size_t i = 0; i < b.size(); i++) a[i] -= b[i];
  return a;
}
poly operator*=(poly& a, const mint& k) {
  if (k == 1) return a;
  for (size_t i = 0; i < a.size(); i++) a[i] *= k;
  return a;
}
poly operator/=(poly& a, const mint& k) { return a *= 1 / k; }
poly operator<<=(poly& a, const int& k) {
  // mnltiple by x^k
  a.insert(a.begin(), k, 0);
  return a;
}
poly operator>>=(poly& a, const int& k) {
  // divide by x^k
  a.erase(a.begin(), a.begin() + min(k, (int)a.size()));
  return a;
}
poly operator*(const poly& a, const poly& b) {
  if (a.empty() || b.empty()) return {};
  int rlen = a.size() + b.size() - 1;
  int len = glim(rlen);
```

```
  if (1ull * a.size() * b.size() <= 1ull * len * bitctz(len)) {
    poly ret(rlen);
    for (size_t i = 0; i < a.size(); i++)
      for (size_t j = 0; j < b.size(); j++) ret[i + j] += a[i] * b[j];
    return ret;
  } else {
    return concalc(len, {a, b},
                   [](vector<mint> vec) { return vec[0] * vec[1]; })
        .cut(rlen);
  }
}
poly operator/(poly a, poly b) {
  if (a.size() < b.size()) return {};
  int rlen = a.size() - b.size() + 1;
  reverse(a.begin(), a.end());
  reverse(b.begin(), b.end());
  a = (a * getInv(b, rlen)).cut(rlen);
  reverse(a.begin(), a.end());
  return a;
}
poly operator-(poly a, const poly& b) { return a -= b; }
poly operator%(const poly& a, const poly& b) {
  return (a - (a / b) * b).cut(b.size() - 1);
}
poly operator*=(poly& a, const poly& b) { return a = a * b; }
poly operator/=(poly& a, const poly& b) { return a = a / b; }
poly operator%=(poly& a, const poly& b) { return a = a % b; }
poly operator+(poly a, const poly& b) { return a += b; }
poly operator*(poly a, const mint& k) { return a *= k; }
poly operator*(const mint& k, poly a) { return a *= k; }
poly operator/(poly a, const mint& k) { return a /= k; }
poly operator<<(poly a, const int& k) { return a <<= k; }
poly operator>>(poly a, const int& k) { return a >>= k; }
poly getDev(poly a) {
  a >>= 1;
  for (size_t i = 1; i < a.size(); i++) a[i] *= i + 1;
  return a;
}
poly getInt(poly a) {
  a <<= 1;
  for (size_t i = 1; i < a.size(); i++) a[i] /= i;
  return a;
}
poly getLn(const poly& a, int lim) {
  assert(a[0] == 1);
  return getInt(getDev(a) * getInv(a, lim)).cut(lim);
}
poly getExp(const poly& a, int lim) {
  assert(a[0] == 0);
  poly b{1};
  for (int len = 2; len <= glim(lim); len <<= 1) {
    poly c = vector<mint>(a.begin(), a.begin() + min(len, (int)a.size()));
    b = concalc(len << 1, {b, getLn(b, len), c}, [](vector<mint> vec) {
          return vec[0] * (1 - vec[1] + vec[2]);
```

```
        }).cut(len);
    }
    return b.cut(lim);
}
poly qpow(const poly& a, string k, int lim) {
    size_t i = 0;
    while (i < a.size() && a[i] == 0) i += 1;
    if (i == a.size() || (i > 0 && k.size() >= 9) ||
        1ull * i * raw(mint(k)) >= 1ull * lim)
        return poly(lim);
    lim -= i * raw(mint(k));
    return getExp(getLn(a / a[i] >> i, lim) * k, lim) *
                qpow(a[i], raw(modint<mint::mod - 1>(k)))
           << i * raw(mint(k));
}
poly qpow(const poly& a, LL k, int lim) {
    size_t i = 0;
    while (i < a.size() && a[i] == 0) i += 1;
    if (i == a.size() || (i > 0 && k >= 1e9) ||
        1ull * i * k >= 1ull * lim)
        return poly(lim);
    lim -= i * k;
    return getExp(getLn(a / a[i] >> i, lim) * k, lim) *
                qpow(a[i], raw(modint<mint::mod - 1>(k)))
           << i * k;
}
mint sqrt(const mint& c) {
    static const auto check = [](mint c) {
        return qpow(c, (mint::mod - 1) >> 1) == 1;
    };
    if (raw(c) <= 1) return 1;
    if (!check(c)) throw "No solution!";
    static mt19937 rng{random_device{}()};
    mint a = rng();
    while (check(a * a - c)) a = rng();
    typedef pair<mint, mint> number;
    const auto mul = [=](number x, number y) {
        return make_pair(x.first * y.first + x.second * y.second * (a * a - c),
                         x.first * y.second + x.second * y.first);
    };
    const auto qpow = [=](number a, int b) {
        number r = {1, 0};
        for (; b; b >>= 1, a = mul(a, a))
            if (b & 1) r = mul(r, a);
        return r;
    };
    mint ret = qpow({a, 1}, (mint::mod + 1) >> 1).first;
    return min(raw(ret), raw(-ret));
}
poly getSqrt(const poly& a, int lim) {
    poly b{sqrt(a[0])};
    for (int len = 2; len <= glim(lim); len <<= 1) {
        poly c = vector<mint>(a.begin(), a.begin() + min(len, (int)a.size()));
        b = (c * getInv(b * 2, len) + b / 2).cut(len);
```

```
  }
  return b.cut(lim);
}
template <class T>
mint divide_at(poly f, poly g, T n) {
  for (; n; n >>= 1) {
    poly r = g;
    for (size_t i = 1; i < r.size(); i += 2) r[i] *= -1;
    f *= r;
    g *= r;
    int i;
    for (i = n & 1; i < (int)f.size(); i += 2) f[i >> 1] = f[i];
    f.resize(i >> 1);
    for (i = 0; i < (int)g.size(); i += 2) g[i >> 1] = g[i];
    g.resize(i >> 1);
  }
  return f.empty() ? 0 : f[0] / g[0];
}
template <class T>
mint linear_rec(poly a, poly f, T n) {
  // a[n] = sum_i f[i] * a[n - i]
  a.resize(f.size() - 1);
  f = poly{1} - f;
  poly g = a * f;
  g.resize(a.size());
  return divide_at(g, f, n);
}
poly BM(poly a) {
  poly ans, lst;
  int w = 0;
  mint delta = 0;
  for (size_t i = 0; i < a.size(); i++) {
    mint tmp = -a[i];
    for (size_t j = 0; j < ans.size(); j++) tmp += ans[j] * a[i - j - 1];
    if (tmp == 0) continue;
    if (ans.empty()) {
      w = i;
      delta = tmp;
      ans = vector<mint>(i + 1, 0);
    } else {
      auto now = ans;
      mint mul = -tmp / delta;
      if (ans.size() < lst.size() + i - w) ans.resize(lst.size() + i - w);
      ans[i - w - 1] -= mul;
      for (size_t j = 0; j < lst.size(); j++) ans[i - w + j] += lst[j] * mul;
      if (now.size() <= lst.size() + i - w) {
        w = i;
        lst = now;
        delta = tmp;
      }
    }
  }
  return ans << 1;
}
```

```
poly lagrange(const vector<pair<mint, mint>>& a) {
  poly ans(a.size()), product{1};
  for (size_t i = 0; i < a.size(); i++) {
    product *= poly{-a[i].first, 1};
  }
  auto divide2 = [&](poly a, mint b) {
    poly res(a.size() - 1);
    for (size_t i = (int)a.size() - 1; i >= 1; i--) {
      res[i - 1] = a[i];
      a[i - 1] -= a[i] * b;
    }
    return res;
  };
  for (size_t i = 0; i < a.size(); i++) {
    mint denos = 1;
    for (size_t j = 0; j < a.size(); j++) {
      if (i != j) denos *= a[i].first - a[j].first;
    }
    poly numes = divide2(product, -a[i].first);
    ans += a[i].second / denos * numes;
  }
  return ans;
}
}
using namespace polystd;"\n"
```

PolynomialMultiplier.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

const int MAXN = 3 * 1e6 + 10, P = 998244353, G = 3, Gi = 332748118;
class PolynomialMultiplier {
public:
    char buf[1 << 21], *p1 = buf, *p2 = buf;
    int N, M, limit = 1, L;
    int r[MAXN];
    long long a[MAXN], b[MAXN];

    inline long long fastPow(long long a, long long k) {
        long long base = 1;
        while(k) {
            if(k & 1) base = (base * a) % P;
            a = (a * a) % P;
```

```
            k >>= 1;
        }
        return base % P;
    }

    inline void NTT(long long *A, int type) {
        for(int i = 0; i < limit; i++)
            if(i < r[i]) swap(A[i], A[r[i]]);
        for(int mid = 1; mid < limit; mid <<= 1) {
            long long Wn = fastPow(type == 1 ? G : Gi, (P - 1) / (mid << 1));
            for(int j = 0; j < limit; j += (mid << 1)) {
                long long w = 1;
                for(int k = 0; k < mid; k++, w = (w * Wn) % P) {
                    int x = A[j + k], y = w * A[j + k + mid] % P;
                    A[j + k] = (x + y) % P,
                    A[j + k + mid] = (x - y + P) % P;
                }
            }
        }
        if(type == -1) {
            long long inv = fastPow(limit, P - 2);
            for(int i = 0; i < limit; i++) A[i] = A[i] * inv % P;
        }
    }

public:
    PolynomialMultiplier() {
        // 初始化一些必要的变量
        for(int i = 0; i < MAXN; i++) r[i] = 0;
        for(int i = 0; i < MAXN; i++) a[i] = 0;
        for(int i = 0; i < MAXN; i++) b[i] = 0;
    }

    vector<int> multiply(vector<int> &A, vector<int> &B) {
        N = A.size() - 1;
        M = B.size() - 1;
        copy(A.begin(), A.end(), a);
        copy(B.begin(), B.end(), b);

        while(limit <= N + M) limit <<= 1, L++;
        for(int i = 0; i < limit; i++) r[i] = (r[i >> 1] >> 1) | ((i & 1) << (L -
1));

        NTT(a, 1);
        NTT(b, 1);
        for(int i = 0; i < limit; i++) a[i] = (a[i] * b[i]) % P;
        NTT(a, -1);

        vector<int> result(N + M + 1);
        for(int i = 0; i <= N + M; i++) result[i] = a[i] % P;

        return result;
    }
};
```

```cpp
// 任意模数 NTT
ll mod_pow(ll a, ll e, ll mod) {
    ll r = 1;
    while (e) {
        if (e & 1) r = (__int128)r * a % mod;
        a = (__int128)a * a % mod;
        e >>= 1;
    }
    return r;
}
struct NTT {
    ll mod;
    ll root;
    NTT() {}
    NTT(ll m, ll g): mod(m), root(g) {}
    void ntt(vector<ll>& a, bool invert) {
        int n = a.size();
        int j = 0;
        for (int i = 1; i < n; i++) {
            int bit = n >> 1;
            for (; j & bit; bit >>= 1) j ^= bit;
            j ^= bit;
            if (i < j) swap(a[i], a[j]);
        }
        for (int len = 2; len <= n; len <<= 1) {
            ll wlen = mod_pow(root, (mod - 1) / len, mod);
            if (invert) wlen = mod_pow(wlen, mod - 2, mod);
            for (int i = 0; i < n; i += len) {
                ll w = 1;
                for (int j = 0; j < len/2; j++) {
                    ll u = a[i+j];
                    ll v = (ll)((__int128)a[i+j+len/2] * w % mod);
                    a[i+j] = u + v;
                    if (a[i+j] >= mod) a[i+j] -= mod;
                    a[i+j+len/2] = u - v;
                    if (a[i+j+len/2] < 0) a[i+j+len/2] += mod;
                    w = (ll)((__int128)w * wlen % mod);
                }
            }
        }
        if (invert) {
            ll inv_n = mod_pow(n, mod - 2, mod);
            for (int i = 0; i < n; i++) a[i] = (ll)((__int128)a[i] * inv_n % mod);
        }
    }
    vector<ll> multiply(const vector<ll>& a, const vector<ll>& b) {
        if (a.empty() || b.empty()) return {};
        int rlen = (int)a.size() + (int)b.size() - 1;
        int n = 1;
        while (n < rlen) n <<= 1;
        vector<ll> fa(n), fb(n);
        for (size_t i = 0; i < a.size(); i++) fa[i] = a[i] % mod;
```

```
        for (size_t i = 0; i < b.size(); i++) fb[i] = b[i] % mod;
        ntt(fa, false);
        ntt(fb, false);
        for (int i = 0; i < n; i++) fa[i] = (ll)((__int128)fa[i] * fb[i] % mod);
        ntt(fa, true);
        fa.resize(rlen);
        return fa;
    }
};

ll inv_mod(ll a, ll m) {
    ll b = m, x = 1, y = 0;
    while (b) {
        ll q = a / b;
        ll t = a - q * b; a = b; b = t;
        t = x - q * y; x = y; y = t;
    }
    x %= m;
    if (x < 0) x += m;
    return x;
}

vector<ll> convolution_mod(const vector<ll>& a, const vector<ll>& b, ll mod) {
    const ll m1 = 167772161; // 5 * 2^25 + 1, root = 3
    const ll m2 = 469762049; // 7 * 2^26 + 1, root = 3
    const ll m3 = 1224736769; // 73 * 2^24 + 1, root = 3
    static NTT ntt1(m1, 3), ntt2(m2, 3), ntt3(m3, 3);
    auto c1 = ntt1.multiply(a, b);
    auto c2 = ntt2.multiply(a, b);
    auto c3 = ntt3.multiply(a, b);
    int rlen = c1.size();
    vector<ll> res(rlen);
    ll m1_mod_m2 = m1 % m2;
    ll m1m2_mod_m3 = ( (__int128)m1 % m3) * (m2 % m3) % m3;
    ll inv_m1_mod_m2 = inv_mod(m1_mod_m2, m2);
    ll inv_m12_mod_m3 = inv_mod(m1m2_mod_m3, m3);
    for (int i = 0; i < rlen; i++) {
        ll x1 = c1[i];
        ll x2 = c2[i];
        ll x3 = c3[i];
        ll t1 = x1;
        ll t2 = ( (__int128)(x2 - t1) % m2 + m2 ) % m2;
        t2 = ( (__int128)t2 * inv_m1_mod_m2 ) % m2;
        ll t3 = ( (__int128)(x3 - (t1 + (__int128)t2 * m1) % m3) % m3 + m3 ) % m3;
        t3 = ( (__int128)t3 * inv_m12_mod_m3 ) % m3;
        __int128 value = (__int128)t1 + (__int128)t2 * m1 + (__int128)t3 * m1 *
m2;
        ll finalv = (ll)(value % mod);
        if (finalv < 0) finalv += mod;
        res[i] = finalv;
    }
    return res;
}
```

```
void solve() {
    int n, m;
    ll p;
    cin >> n >> m >> p;
    vector<ll> A(n+1), B(m+1);
    for (int i = 0; i <= n; i++) cin >> A[i];
    for (int i = 0; i <= m; i++) cin >> B[i];
    auto C = convolution_mod(A, B, p);
    for (int i = 0; i <= n + m; i++) {
        if (i) cout << " ";
        cout << (C[i] % p + p) % p;
    }
    cout << "\n";
}

signed main() {
#if DEBUG
    freopen("input.txt", "r", stdin);
    auto start_time = chrono::steady_clock::now();
#else
    ios::sync_with_stdio(false);
#endif
    cin.tie(nullptr);

    int t = 1;

    while (t--) {
        solve();
    }

#if DEBUG
    auto end_time = chrono::steady_clock::now();
    auto diff = chrono::duration_cast<chrono::milliseconds>(end_time -
start_time);
    cerr << "Time: " << diff.count() << " ms" << endl;
#endif

    return 0;
}
"\n"
```

Primes.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define DEBUG 1
const ll N = 2000000;

ll qpow(ll base,ll k,ll mod) {
    ll res = 1;
```

```
    if (k < 0) {
        return qpow(qpow(base,-k,mod),mod-2,mod);
    }
    while (k) {
        if (k & 1) {
            res *= base; res %= mod;
        }
        k >>= 1;
        base *= base; base %= mod;
    }
    return res;
}

ll phi(ll x) {
    ll y = x;
    vector<ll> q;
    for (int i = 2;i*i<=y;i++) {
        if (y % i == 0) {
            q.push_back(i);
        }

        while (y % i == 0) {
            y /= i;
        }
    }

    if (y > 1) q.push_back(y);

    for (auto v : q) {
        x /= v;
        x *= (v - 1);
    }

    return x;
}

class Primes{
public:
    ll notPrime[N];
    ll phi[N],mu[N];
    vector<ll> primes;
    void sieve(int maxn){
        phi[1] = 1;
        mu[1] = 1;
        for (ll i = 2;i<maxn + 1;i++){
            if (!notPrime[i]){
                primes.push_back(i);
                phi[i] = i - 1;
                mu[i] = -1;
            }

            for (auto p : primes) {
                if (i * p > maxn) break;
                notPrime[i * p] = 1;
```

```
                if (i % p == 0) {
                    phi[i * p] = phi[i] * p;
                    mu[i * p] = 0;
                    break;
                }
                phi[i * p] = phi[i] * phi[p];
                mu[i * p] = -mu[i];
            }
        }
        return;
    }
};

// 找奇质数 x 的原根,一定是奇素数!!!
// 时间复杂度不详
ll getYuanGen(ll x) {
    Primes solver;
    solver.sieve(x);
    ll y = x - 1;
    vector<ll> q;
    for (int i = 0;i<solver.primes.size() && y > 1;i++) {
        if (y % solver.primes[i] == 0) {
            q.push_back(solver.primes[i]);
        }
        while (y % solver.primes[i] == 0) {
            y /= solver.primes[i];
        }
    }

    for (ll i = 1;i<x;i++) {
        bool ok = 1;
        for (auto v : q) {
            if (qpow(i,(x-1)/v,x) == 1) {
                ok = 0;
                break;
            } else {

            }
        }
        if (ok) return i;
    }

    return 0;
}"\n"
```

Pythagoras.cpp

```
#include <bits/stdc++.h>
using namespace std;
/*
用于寻找基元勾股数
```

```
*/
typedef long long ll;
class Pythagoras{
public:
    vector<tuple<int,int,int>> q;
    int maxN;
    Pythagoras(int MAXN){
        maxN = MAXN;
    }
    void init(){
        for (int i = 1;i<=maxN;i++){
            for (int j = i+1;2*i*j<=maxN && j*j+i*i<=maxN;j++){
                ll a = j * j - i * i;
                ll b = 2 * i * j;
                ll c = j * j + i * i;
                if (b > c){
                    swap(b,c);
                }
                if (a > b){
                    swap(a,b);
                }
                if (__gcd(a,b) == 1){
                    for (int k = 1;k*c<=maxN;k++){
                        q.emplace_back(a*k,b*k,c*k);
                    }
                }
            }
        }
        return;
    }
};"\n"
```

Random.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

class RandomNumberGenerator{
public:
    RandomNumberGenerator() : gen(std::random_device{}()){}

    ll generate(ll l,ll r){
        uniform_int_distribution<> dis(l,r);
        return dis(gen);
    }
private:
    mt19937 gen;
};"\n"
```

WheelSieve.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
using ll = long long;
using pii = pair<int,int>;
using i128 = __int128_t;
using pll = pair<ll,ll>;
const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

/*
2025牛客多校期间从 wanna be free 处 cv 得到的板子
可以快速(1800ms)筛出 1e9 以内质数
调用时,需要指定 U 表示筛选的最大质数的大小,可以输入 1e9
然后 init ,之后从 1 开始, prime 数组会填充质数, 遍历到为 0 的位置就表示没有质数了
1e9 以内质数大概有 5e7 个!!
*/

namespace wheelSieve {
    using std::cin,std::cout;
    using std::max,std::memcpy;

    #define N 50847534
    #define block 510510
    #define block_size 15953
    #define M 7
    #define K 1959

    #define set(a, b) a[b >> 5] |= 1 << (b & 31)
    typedef unsigned int uint;
    typedef unsigned char uchar;

    uint prime[N + 7], pre_block[block_size + 7], cur_block[block_size + 7];
    uchar p[block + 7];
    int U;
    void init(){
        uint cnt = 0;
        p[0] = p[1] = true;
        set(pre_block, 0);
        set(pre_block, block);
        for ( uint i = 2; i <= block; ++i){
            if (!p[i]){
                prime[++cnt] = i;
                if (cnt <= M) set(pre_block, i);
            }
            for ( uint j = 1; j <= cnt && i * prime[j] <= block; ++j){
                uint t = i * prime[j];
                p[t] = true;
                if (j <= M) set(pre_block, t);
```

```
                if (i % prime[j] == 0) break;
            }
        }
        for ( uint i = 1, j = cnt; i < K; ++i){
            uint end = (i + 1) * block - 1, start = i * block;
            memcpy(cur_block, pre_block, sizeof(cur_block));
            for ( uint k = M + 1; prime[k] * prime[k] <= end; ++k){
                uint t1 = max((start - 1) / prime[k] + 1, prime[k]) * prime[k], t2
= prime[k] << 1;
                for ( uint l = (t1 & 1 ? t1 : t1 + prime[k]) - start; l < block; l
+= t2){
                    set(cur_block, l);
                }
            }
            for ( uint k = 0; k <= block_size; ++k){
                uint t1 = ~cur_block[k];
                while (t1){
                    uint t2 = __builtin_ctz(t1);
                    if ((k << 5) + t2 >= block) break;
                    prime[++j] = start + (k << 5) + t2;
                    if (j >= N||prime[j]>=U) return;
                    t1 -= t1 & ((~t1) + 1);
                }
            }
        }
    }
};"\n"
```