

# Summary

---

Run summarize.bat to update this file.

Last Updated: 2025-12-09 15:36:09

Author: HuangZy

[TOC]

## ST表.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
#define FUCK if (DEBUG) cout << "fuck" << endl;
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 = 1e18;
const ll MOD = 1e9 + 7;

/*
使用前记得 init
*/
namespace ST {
    int lg[N];
    void init(int n) {
        lg[0] = 0, lg[1] = 0;
        for (int i = 2; i <= n; i++) {
            lg[i] = lg[i / 2] + 1;
        }
    }
    struct ST {
        struct Info {
            ll val;
            Info() {
                val = INF;
            }
            Info operator+(const Info& A) const {
                Info z;
                z.val = min(val, A.val);
                return z;
            }
        };
        vector<vector<Info>> f;
        void init(const vector<ll>& a) {
            int n = (int)a.size();
```

```

        if (n == 0) return;
        for (int i = 2; i <= n; ++i) lg[i] = lg[i >> 1] + 1;
        int K = lg[n];
        f.assign(K + 1, vector<Info>(n));
        for (int i = 0; i < n; ++i) f[0][i].val = a[i];
        for (int k = 1; k <= K; ++k)
            for (int i = 0; i + (1 << k) <= n; ++i)
                f[k][i] = f[k - 1][i] + f[k - 1][i + (1 << (k - 1))];
    }
    Info query(int l, int r) {
        if (l > r) return Info();
        int len = r - l + 1;
        int k = lg[len];
        return f[k][l] + f[k][r - (1 << k) + 1];
    }
};
}

```

## 伸展树.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;
}
};

```

## 左偏树.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;
}
};

```

## 并查集.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;
    }
};

```

## 李超树.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
*/
struct LiChaoTree {
    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);
}

};

/*
应对 db 的版本
init 函数中给出定义域即可
long double 比 double 慢得多, 慎用!!!
*/

using db = double;
const db INF_LD = numeric_limits<db>::infinity();

struct LiChao {
    struct Line { db m, b; };
    struct Node { Line ln; Node *l, *r; Node(Line v):ln(v),l(nullptr),r(nullptr){}
};

```

```

db L, R;
Node *root;
int maxDepth;
LiChao(db _L = 0, db _R = 0, int _maxDepth =
50):L(_L),R(_R),root(nullptr),maxDepth(_maxDepth){}
inline db eval(const Line &ln, db x) const { return ln.m * x + ln.b; }
void addLine(Line nw, Node *&nd, db l, db r, int depth) {
    if (!nd) { nd = new Node(nw); return; }
    db m = (l + r) / 2;
    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 (depth == 0) return;
    if (lef != mid) addLine(nw, nd->l, l, m, depth-1);
    else addLine(nw, nd->r, m, r, depth-1);
}
void addLine(db m, db b) { addLine({m,b}, root, L, R, maxDepth); }
db query(db x, Node *&nd, db l, db r, int depth) const {
    if (!nd) return INF_LD;
    db res = eval(nd->ln, x);
    if (depth == 0) return res;
    db mid = (l + r) / 2;
    if (x <= mid) return min(res, query(x, nd->l, l, mid, depth-1));
    else return min(res, query(x, nd->r, mid, r, depth-1));
}
db query(db x) const { return query(x, root, L, R, maxDepth); }
void init(db _L, db _R, int _maxDepth = 50) {
    L = _L;
    R = _R;
    maxDepth = _maxDepth;
    root = nullptr;
}
};

```

## 树上启发式合并.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;
    };

};

```

## 树状数组.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);
}
};

```

## 珂朵莉树.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;
}
} ;

```

## 笛卡尔树.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;
}
};

```

## 线段树.cpp

```

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

/*
切记 pd 清空旧标记
*/

#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 q,int val){
        if (l > q || r < q) return;
        if (l == r) {
            return;
        }
        int mid = l+r>>1;
        pd(rt);
        if (q <= mid){
            modify(LS,l,mid,q,val);
        }
        if (q >= mid + 1){
            modify(RS,mid+1,r,q,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;

```

## 高精度.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;
    }
};
```



```

        }
    }
    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;
}

```

## 三角形.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;
    }
};

```

## 凸包.cpp

```

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

/*
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.
注意下标从 1 开始
*/

template<typename T>
struct Point {
    T x, y;
    int idx;
    Point(T X = 0, T 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);
    }

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

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

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

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

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

```

```

template<typename T, int N>
class ConvexHull {
public:
    int n;
    bool vis[N];
    Point<T> points[N], hull[N];
    static bool cmp_y(const Point<T>& A, const Point<T>& B) {
        if (A.y == B.y){
            return A.x < B.x;
        }
        return A.y < B.y;
    }

    static bool cmp_sita(const Point<T>& A, const Point<T>& B, const Point<T>&
base) {
        Point<T> A_base = A - base;
        Point<T> 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<T>& A,
const Point<T>& 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];
        }
    }
}
}

double len(Point<T> x) {
    return sqrt(x.x * x.x + x.y * x.y);
}

double getS() {
    double ans = 0;
    for (int i = 2; i+1<=tp; i++) {
        ans += ((hull[i] - hull[1]) ^ (hull[i + 1] - hull[1]));
    }
    ans = abs(ans) / 2;
    return ans;
}

double getC() {
    double ans = 0;
    for (int i = 1; i+1<=tp; i++) {
        ans += len(hull[i + 1] - hull[i]);
    }
    ans += len(hull[1] - hull[tp]);
    return ans;
}
} ;

```

## 圆的并.cpp

```

#define DEBUG 1
#define FUCK cout << "fuck" << endl;
#ifdef DEBUG
    #include "all.hpp"
#else
    #include <bits/stdc++.h>
#endif

using namespace std;
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;

```

```

using db = long double;
using pdd = pair<db, db>;
using i128 = __int128_t;

const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

namespace CircleUnion {
    const db PI = acos(-1);
    const db eps = 2e-10, pi2 = PI * 2.0;
    struct Circle { db x, y, r; };

    enum relation { outside = 0, interseptive = 1, contained = 2, containing = 3 };

    struct Vec { db x, y; Vec(db _x=0, db _y=0):x(_x),y(_y){} };

    inline db sq(db x){return x*x;}
    inline db f(const Vec &O, db r, db t){ return r * (r * t + O.x * sin(t) - O.y
* cos(t)); }

    relation circle_relation(const Vec &O1, db r1, const Vec &O2, db r2) {
        db d2 = sq(O1.x - O2.x) + sq(O1.y - O2.y);
        if ((r1 + r2) * (r1 + r2) <= d2 + eps) return outside;
        if ((r1 - r2) * (r1 - r2) >= d2 - eps) return (r1 <= r2 + eps) ? contained
: containing;
        return interseptive;
    }

    void intersection(const Vec &O1, db r1, const Vec &O2, db r2, db &beg, db
&end) {
        Vec O12{O2.x - O1.x, O2.y - O1.y};
        db d2 = sq(O12.x) + sq(O12.y), d = sqrt(d2);
        db Cos = (((r1 + r2) * (r1 - r2) + d2) / (2.0 * d * r1));
        if (Cos > 1) Cos = 1; if (Cos < -1) Cos = -1;
        db sAng = acos(Cos), iAng = atan2(O12.y, O12.x);
        if (iAng < 0.0) iAng += pi2;
        beg = iAng - sAng; if (beg < 0.0) beg += pi2;
        end = iAng + sAng; if (end >= pi2) end -= pi2;
    }

    static db union_area_ = 0.0, union_perim_ = 0.0;

    void init(const vector<Circle> &circles) {
        int n = (int)circles.size();
        vector<Vec> O(n); vector<db> r(n);
        for (int i=0;i<n;++i){ O[i].x = circles[i].x; O[i].y = circles[i].y; r[i]
= circles[i].r; }
        union_area_ = union_perim_ = 0.0;
        for (int i = 0; i < n; ++i) {
            vector<pair<db,db>> segs; segs.reserve(n*2+2);
            bool skip = false;
            for (int j = 0; j < n; ++j) {

```

```

        relation rel = circle_relation(O[i], r[i], O[j], r[j]);
        if (rel == contained) {
            if (fabs(r[i] - r[j]) > eps || i > j) { skip = true; break; }
        }
    }
    if (skip) continue;
    for (int j = 0; j < n; ++j) {
        relation rel = circle_relation(O[i], r[i], O[j], r[j]);
        if (rel == intersective) {
            db a,b; intersection(O[i], r[i], O[j], r[j], a, b);
            if (a <= b) segs.emplace_back(a,b);
            else { segs.emplace_back(0.0, b); segs.emplace_back(a, pi2); }
        }
    }
    segs.emplace_back(pi2, pi2);
    sort(segs.begin(), segs.end(), [](const pair<db,db>&A,const
pair<db,db>&B){ if (A.first!=B.first) return A.first < B.first; return A.second <
B.second; });
    db la = 0.0;
    for (size_t k = 0; k < segs.size(); ++k) {
        db L = segs[k].first, R = segs[k].second;
        if (la < L) {
            union_area_ += 0.5 * (f(O[i], r[i], L) - f(O[i], r[i], la));
            union_perim_ += r[i] * (L - la);
            la = R;
        } else if (la < R) {
            la = R;
        }
    }
}

double area(){ return union_area_; }
double perimeter(){ return union_perim_; }
}
using namespace CircleUnion;

```

## 平面最近点对.cpp

```

#include <bits/stdc++.h>
using namespace std;
using db = double;
using i128 = __int128;

template<typename T>
auto closest_pair_sq(vector<pair<T,T>>& a){
    using R = conditional_t<is_integral_v<T>, long long, db>;
    if(a.size()<2) return R(0);
    sort(a.begin(), a.end(), [](auto&p,auto&q){ return p.first<q.first; });
    vector<pair<T,T>> buf(a.size());

```

```

function<R(int,int)> rec = [&](int l,int r)->R{
    if(r-l<=1) return numeric_limits<R>::max();
    int m=(l+r)/2; T midx=a[m].first;
    R d = min(rec(l,m), rec(m,r));

    merge(a.begin()+l, a.begin()+m, a.begin()+m, a.begin()+r,
          buf.begin(), [](auto&p,auto&q){ return p.second<q.second; });
    copy(buf.begin(), buf.begin()+(r-l), a.begin()+l);

    vector<pair<T,T>> v; v.reserve(r-l);

    if constexpr(is_integral_v<T>){
        i128 D = d;
        for(int i=l;i<r;++i){
            i128 dx = (i128)a[i].first - midx;
            if(dx*dx <= D) v.push_back(a[i]);
        }
        for(int i=0;i<(int)v.size();++i)
            for(int j=i+1;j<(int)v.size() &&
                (i128)(v[j].second-v[i].second)*(v[j].second-v[i].second) <=
D; ++j){
                i128 dx = (i128)(v[j].first - v[i].first);
                i128 dy = (i128)(v[j].second - v[i].second);
                i128 w = dx*dx + dy*dy;
                if(w < D) D = w;
            }
        return (R)D;
    } else {
        db D = d;
        for(int i=l;i<r;++i){
            db dx = db(a[i].first) - db(midx);
            if(dx*dx <= D) v.push_back(a[i]);
        }
        for(int i=0;i<(int)v.size();++i)
            for(int j=i+1;j<(int)v.size() &&
                db(v[j].second-v[i].second)*db(v[j].second-v[i].second) <= D;
++j){
                db dx = db(v[j].first - v[i].first);
                db dy = db(v[j].second - v[i].second);
                db w = dx*dx + dy*dy;
                if(w < D) D = w;
            }
        return (R)D;
    }
};

return rec(0, (int)a.size());
}

```



```

#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;

```

## 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;

```

## Tarjan.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;

// 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边，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;

```

## 图.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;
    }
};

```

## 斯坦纳树.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;
}
};

```

## 最短路.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];
        }
    }
}

```

```
    }
};
```

## 树哈希.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
#define endl '\n'
#define FUCK if (DEBUG) cout << "fuck" << endl;
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 = 1e18;
const ll MOD = 1e9 + 7;

/*
定义 val[u] = 1 + f(h(val[v]))
v is son of u
*/

ll h(ll x) {
    return x * x * x * 1237123 + 19260817;
}
ll f(ll x) {
    ll cur = h(x & ((1 << 31) - 1)) + h(x >> 31);
    return cur;
}
```

## 树链剖分.cpp

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

#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 q,int val){
    if (l > q || r < q) return;
    if (l == r) {
        return;
    }
    int mid = l+r>>1;
    pd(rt);
    if (q <= mid){
        modify(LS,l,mid,q,val);
    }
    if (q >= mid + 1){
        modify(RS,mid+1,r,q,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));
        }
    }
};

struct TreeChainSeg {
    vector<vector<int>> g;
    SegTree seg;
    vector<int> fa, top, siz, son, dep, dfn, id;
    int cnt, n, cnt2;

    void dfs1(int u) {
        siz[u] = 1;
        for (auto v : g[u]) {
            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;
            }
        }
        return;
    }

    void dfs2(int u, int tp) {
        id[cnt] = u;
        dfn[u] = ++cnt;
        top[u] = tp;
        if (son[u]) {
            dfs2(son[u], tp);
        }
        for (auto v : g[u]) {
            if (v == fa[u] || v == son[u]) continue;
            dfs2(v, v);
        }
        return;
    }

    void init(vector<vector<int>>& G) {
        g = G;
        n = g.size();
        cnt = 0, cnt2 = n + 1;
        fa.clear(); fa.resize(n + 1, 0);
        top.clear(); top.resize(n + 1, 0);
        siz.clear(); siz.resize(n + 1, 0);
    }
};

```

```

        son.clear(); son.resize(n + 1, 0);
        dep.clear(); dep.resize(n + 1, 0);
        dfn.clear(); dfn.resize(n + 1, 0);
        id.clear(); id.resize(n + 1, 0);

        dfs1(1), dfs2(1, 1);
        seg.build(1, 1, n);
        return;
    }

    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;
        }
        return u;
    }
};

```

## 流.cpp

```

#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
#define FUCK if (DEBUG) cout << "fuck" << endl;
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;

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;
    }
};

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 };
}

};

// 有源汇上下界最大流
// add(S, E, L, R) 插入边
// solve(S, E) 输出最大流, -1 表示无解!!!
namespace MCMF{
    const int MAXN = 100000 + 3;
    const int MAXM = 700000 + 3; // 增大上界, 避免边数超界
    int H[MAXN], V[MAXM], N[MAXM];
    ll F[MAXM]; // 改为 ll
    int o = 1, n;
    void add0(int u, int v, ll f){
        V[++ o] = v; N[o] = H[u]; H[u] = o; F[o] = f;
        V[++ o] = u; N[o] = H[v]; H[v] = o; F[o] = 0;
        n = max(n, u);
        n = max(n, v);
    }
    ll D[MAXN];
    bool bfs(int s, int t){
        queue<int> Q;
        for(int i = 1; i <= n; ++ i) D[i] = 0;
        Q.push(s); D[s] = 1;
        while(!Q.empty()){
            int u = Q.front(); Q.pop();
            for(int i = H[u]; i; i = N[i]){
                int v = V[i];
                ll cap = F[i];
                if(cap != 0 && !D[v]){
                    D[v] = D[u] + 1;
                    Q.push(v);
                }
            }
        }
        return D[t] != 0;
    }
    int C[MAXN];
    ll dfs(int s, int t, int u, ll maxf){
        if(u == t) return maxf;
        ll totf = 0;
        for(int &i = C[u]; i; i = N[i]){
            int v = V[i];
            ll cap = F[i];
            if(cap && D[v] == D[u] + 1){
                ll pushed = dfs(s, t, v, min(F[i], maxf));

```

```

        if(pushed == 0) continue;
        F[i] -= pushed;
        F[i ^ 1] += pushed;
        totf += pushed;
        maxf -= pushed;
        if(maxf == 0) return totf;
    }
}
return totf;
}
ll mcmf(int s, int t){
    ll ans = 0;
    while(bfs(s, t)){
        memcpy(C, H, sizeof(H));
        ans += dfs(s, t, s, (ll)1e18);
    }
    return ans;
}
ll G[MAXN];
void add(int u, int v, ll l, ll r){
    G[v] += l;
    G[u] -= l;
    add0(u, v, r - l);
}
void clear(){
    for(int i = 1; i <= o; ++ i){
        N[i] = 0; F[i] = 0; V[i] = 0;
    }
    for(int i = 1; i <= n; ++ i){
        H[i] = 0; G[i] = 0; C[i] = 0;
    }
    o = 1; n = 0;
}
bool solve(){ // 无源汇可行性检查
    int s = ++ n;
    int t = ++ n;
    ll sum = 0;
    for(int i = 1; i <= n - 2; ++ i){
        if(G[i] < 0) add0(i, t, -G[i]);
        else add0(s, i, G[i]), sum += G[i];
    }
    auto res = mcmf(s, t);
    if(res != sum) return false;
    return true;
}
ll solve(int s0, int t0){ // 有源汇上下界最大流
    add0(t0, s0, (ll)1e18);
    int s = ++ n;
    int t = ++ n;
    ll sum = 0;
    for(int i = 1; i <= n - 2; ++ i){
        if(G[i] < 0) add0(i, t, -G[i]);
        else add0(s, i, G[i]), sum += G[i];
    }
}

```

```

        auto res = mcmf(s, t);
        if(res != sum) return -1;
        return mcmf(s0, t0);
    }
}

```

## 点分治.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;

// 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 res;
    };
    dfs(dfs, 1, 1);
    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;
};

```

## 虚树.cpp

```

#define DEBUG 1
#define FUCK cout << "fuck" << endl;
#ifdef DEBUG
    #include "all.hpp"
#else
    #include <bits/stdc++.h>
#endif

using namespace std;
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
using db = long double;
using pdd = pair<db, db>;

const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

/*
虚树的建立需要依靠 lca，这里写的是 O(1) lca
有时，虚树需要额外插入根节点，才能保证 dp 的正确性
*/

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;

```

```
void solve() {
    int n; cin >> n;
    vector<vector<int>> g(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v; cin >> u >> v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    solver.init(&g, 1);
    vector<int> dfn(n + 1, 0);
    int cntDfn = 0;
    auto dfs0 = [&](auto&&self, int u, int f) -> void {
        dfn[u] = ++cntDfn;
        for (auto v : g[u]) {
            if (v == f) continue;
            self(self, v, u);
        }
    };
    dfs0(dfs0, 1, -1);

    vector<int> spe;
    // 插入特殊点
    sort(spe.begin(), spe.end(), [&](int x, int y){
        return dfn[x] < dfn[y];
    });
    int sz = spe.size();
    for (int i = 0; i + 1 < sz; i++) {
        spe.push_back(solver.lca(spe[i], spe[i + 1]));
    }
    sort(spe.begin(), spe.end(), [&](int x, int y){
        return dfn[x] < dfn[y];
    });
    spe.erase(unique(spe.begin(), spe.end()), spe.end());
    vector<vector<int>> e(n + 1); // 虚树
    for (int i = 0; i + 1 < spe.size(); i++) {
        int lca = solver.lca(spe[i], spe[i + 1]);
        e[lca].push_back(spe[i + 1]);
    }
    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;
}

```

## 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;
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;
}

};

```

## Crt.cpp

/\*

功能简短说明：

1. 输入：两个  $(r, m)$  分别表示  $x \equiv r \pmod{m}$ ；建议  $m > 0$  且  $0 \leq r < m$ 。
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  $x = r + k * M$  ( $k \in \mathbb{Z}$ )，且  $0 \leq r < M$ ；若无解返回  $(-1, -1)$ 。
4. 合并规则：如果方程组相容（即  $r_2 - r_1$  能被  $g = \gcd(m_1, m_2)$  整除），则合并得到模  $M = m_1 * (m_2 / g)$ ，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  $(r_2 - r_1) \% g == 0$ ，计算位移  $t = (r_2 - r_1) / g * x \pmod{m_2 / g}$ ，再算出  $r = (r_1 + m_1 * t) \pmod{M}$ 。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  $x \equiv 8 \pmod{15}$ 。

\*/

```
#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;
}

```

## 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]);
    }
}

using i128 = __int128_t;
// 判断 i128 右移 s 位会不会溢出
bool check (i128 x, int s) {
    if (x == 0) return true;
    unsigned __int128 ux = x < 0 ? -(unsigned __int128)x : (unsigned __int128)x;
    if (s >= 127) return ux == 0;
    unsigned __int128 pos_limit = (((unsigned __int128)1) << 127) - 1 >> s;
    unsigned __int128 neg_limit = (((unsigned __int128)1) << 127) >> s;
    if (x < 0) return ux <= neg_limit;
    else return ux <= pos_limit;
};

```

## 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;
    vector<int> inv;
    void init(int P_) {
        p = P_;
        fact.resize(p);
        inv.resize(p);
        fact[0] = 1;
        for (int i = 1; i < p; i++) {
            fact[i] = 1ll * fact[i - 1] * i % p;
        }
        vector<ll> pre(p, 1), suf(p, 1);
        for (int i = 1; i < p; i++) {
            pre[i] = pre[i - 1] * fact[i] % p;
        }
        suf[p - 1] = p - 1;
        for (int i = p - 2; i >= 1; i--) {
            suf[i] = suf[i + 1] * fact[i] % p;
        }
        ll base = qpow(pre.back(), p - 2, p);
        inv[0] = 1;
        for (int i = 1; i < p; i++) {
            if (i + 1 < p) {
                inv[i] = pre[i - 1] * suf[i + 1] % p;
            } else {
                inv[i] = pre[i - 1];
            }
            inv[i] = inv[i] * base % p;
        }
    }
    int C(int n, int m) {
        if (m > n) return 0;
        return 1ll * fact[n] * inv[n - m] % p * inv[m] % 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;
    }
} solver;

// 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;
}
};

```

## Simpson积分.cpp

```

#define DEBUG 1
#define FUCK cout << "fuck" << endl;
#ifdef DEBUG
    #include "all.hpp"
#else
    #include <bits/stdc++.h>
#endif

using namespace std;
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
using db = long double;
using pdd = pair<db, db>;
using i128 = __int128_t;

const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

// oi-wiki 上扒下来的模版, 异常强大
namespace Simpson {
    db f(db x) {

    }

    double simpson(double l, double r) {
        double mid = (l + r) / 2;
        return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6; // 辛普森公式
    }

    double asr(double l, double r, double eps, double ans, int step) {
        double mid = (l + r) / 2;
        double fl = simpson(l, mid), fr = simpson(mid, r);
        if (abs(fl + fr - ans) <= 15 * eps && step < 0)
            return fl + fr + (fl + fr - ans) / 15; // 足够相似的话就直接返回
        return asr(l, mid, eps / 2, fl, step - 1) +
            asr(mid, r, eps / 2, fr, step - 1); // 否则分割成两段递归求解
    }
}

```

```

    }

    double calc(double l, double r, double eps) {
        return asr(l, r, eps, simpson(l, r), 12);
    }

};

```

## Wheel筛.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);
            }
        }
    }
};

```

## 卡特兰数.cpp

```

/* 死亡回放
你的输入，真的写对了吗?? (2025 ICPC 沈阳 M)
*/
#define DEBUG 1
#define FUCK cout << "fuck" << endl;
#ifdef DEBUG
    #include "all.hpp"
#else
    #include <bits/stdc++.h>
#endif

```

```

using namespace std;
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
using db = long double;
using pdd = pair<db, db>;
using i128 = __int128_t;

const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

namespace ComNum {
    const int MAXN = 5e6;
    ll fac[MAXN], inv[MAXN];

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

    void init() {
        fac[0] = 1;
        for (int i = 1; i < MAXN; i++) {
            fac[i] = fac[i - 1] * i % MOD;
        }
        vector<ll> pre(MAXN, 1), suf(MAXN, 1);
        for (int i = 0; i < MAXN; i++) {
            if (i - 1 >= 0) pre[i] = pre[i - 1];
            pre[i] = pre[i] * fac[i] % MOD;
        }
        for (int i = MAXN - 1; i >= 0; i--) {
            if (i + 1 < MAXN) suf[i] = suf[i + 1];
            suf[i] = suf[i] * fac[i] % MOD;
        }
        ll base = qpow(pre.back(), MOD - 2, MOD);
        for (int i = 0; i < MAXN; i++) {
            ll fenzi = 1;
            if (i - 1 >= 0) fenzi = pre[i - 1] * fenzi % MOD;
            if (i + 1 < MAXN) fenzi = suf[i + 1] * fenzi % MOD;
            inv[i] = fenzi * base % MOD;
        }
        return;
    }
}

```

```

    ll C(ll n, ll m){
        if(m == 0) return 1;
        if(n >= 0){
            if(n < m) return 0;
            return fac[n] * inv[n - m] % MOD * inv[m] % MOD;
        }else{
            ll a = -n;
            ll nn = a + m - 1;
            ll res = fac[nn] * inv[nn - m] % MOD * inv[m] % MOD;
            if(m & 1) res = (MOD - res) % MOD;
            return res;
        }
    }
};

namespace Catalan {
    ll get(int n) {
        ll res = ComNum::C(2 * n, n);
        res = (res - ComNum::C(2 * n, n + 1) + MOD) % MOD;
        return res;
    }
};

```

## 哈希.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;
    }
}

```

## 基元勾股数.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;
}
};

```

## 多项式(旧).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 % mod + str[i] - '0') % mod;
            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;

```



## 多项式.cpp

```
/* 死亡回放
你的输入，真的写对了吗?? (2025 ICPC 沈阳 M)
*/
#define DEBUG 1
#define FUCK cout << "fuck" << endl;
#if DEBUG
    #include "all.hpp"
#else
    #include <bits/stdc++.h>
#endif

using namespace std;
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
using db = long double;
using pdd = pair<db, db>;
using i128 = __int128_t;

const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 998244353;

/*
此模版来自 https://hourai.nanani-fan.club/poly-family/
感谢 哈尔滨工业大学-蓬莱人形 的分享
*/

namespace poly {
    ll qpow(ll base,ll k) {
        if (base == 0) return 0;
        ll res = 1;
        base %= MOD; base = (base + MOD) % MOD;
        k %= (MOD - 1); k = (k + MOD - 1) % (MOD - 1);
        while (k) {
            if (k & 1) {
                res *= base; res %= MOD;
            }
            k >>= 1;
            base *= base; base %= MOD;
        }
        return res;
    }
    ll inv(ll x) {
        return qpow(x, MOD - 2);
    }
}
mt19937 MT;
using Poly = vector<ll>;
#define lg(x) ((x) == 0 ? -1 : __lg(x))
```

```

#define Size(x) int(x.size())
namespace NTT_ns {
const long long G = 3, invG = inv(G);
vector<int> rev;
void NTT(ll* F, int len, int sgn) {
    rev.resize(len);
    for (int i = 1; i < len; ++i) {
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) * (len >> 1));
        if (i < rev[i]) swap(F[i], F[rev[i]]);
    }
    for (int tmp = 1; tmp < len; tmp <= 1) {
        ll w1 = qpow(sgn ? G : invG,
            (MOD - 1) / (tmp << 1));
        for (int i = 0; i < len; i += tmp << 1){
            for(ll j = 0, w = 1; j < tmp; ++j,
                w = w * w1 % MOD) {
                ll x = F[i + j];
                ll y = F[i + j + tmp] * w % MOD;
                F[i + j] = (x + y) % MOD;
                F[i + j + tmp] = (x - y + MOD) % MOD;
            }
        }
    }
    if (sgn == 0) {
        ll inv_len = inv(len);
        for (int i = 0; i < len; ++i)
            F[i] = F[i] * inv_len % MOD;
    }
}
}
using NTT_ns::NTT;
Poly operator * (Poly F, Poly G) {
    int siz = Size(F) + Size(G) - 1;
    int len = 1 << (lg(siz - 1) + 1);
    if (siz <= 300) {
        Poly H(siz);
        for (int i = Size(F) - 1; ~i; --i)
            for (int j = Size(G) - 1; ~j; --j)
                H[i + j] = (H[i + j] + F[i] * G[j])%MOD;
        return H;
    }
    F.resize(len), G.resize(len);
    NTT(F.data(), len, 1), NTT(G.data(), len, 1);
    for (int i = 0; i < len; ++i)
        F[i] = F[i] * G[i] % MOD;
    NTT(F.data(), len, 0), F.resize(siz);
    return F;
}
Poly operator + (Poly F, Poly G) {
    int siz = max(Size(F), Size(G));
    F.resize(siz), G.resize(siz);
    for (int i = 0; i < siz; ++i)
        F[i] = (F[i] + G[i]) % MOD;
    return F;
}

```

```

    }
    Poly operator - (Poly F, Poly G) {
        int siz = max(Size(F), Size(G));
        F.resize(siz), G.resize(siz);
        for (int i = 0; i < siz; ++i)
            F[i] = (F[i] - G[i] + MOD) % MOD;
        return F;
    }
    Poly lsh(Poly F, int k) {
        F.resize(Size(F) + k);
        for (int i = Size(F) - 1; i >= k; --i)
            F[i] = F[i - k];
        for (int i = 0; i < k; ++i) F[i] = 0;
        return F;
    }
    Poly rsh(Poly F, int k) {
        int siz = Size(F) - k;
        for (int i = 0; i < siz; ++i) F[i] = F[i + k];
        return F.resize(siz), F;
    }
    Poly cut(Poly F, int len) {
        return F.resize(len), F;
    }
    Poly der(Poly F) {
        int siz = Size(F) - 1;
        for (int i = 0; i < siz; ++i)
            F[i] = F[i + 1] * (i + 1) % MOD;
        return F.pop_back(), F;
    }
    Poly inte(Poly F) {
        F.emplace_back(0);
        for (int i = Size(F) - 1; ~i; --i)
            F[i] = F[i - 1] * inv(i) % MOD;
        return F[0] = 0, F;
    }
    Poly inv(Poly F) {
        int siz = Size(F); Poly G{inv(F[0])};
        for (int i = 2; (i >> 1) < siz; i <= 1) {
            G = G + G - G * G * cut(F, i), G.resize(i);
        }
        return G.resize(siz), G;
    }
    Poly ln(Poly F) {
        return cut(inte(cut(der(F) * inv(F), Size(F))), Size(F));
    }
    Poly exp(Poly F) {
        int siz = Size(F); Poly G{1};
        for (int i = 2; (i >> 1) < siz; i <= 1) {
            G = G * (Poly{1} - ln(cut(G, i)) + cut(F, i)), G.resize(i);
        }
        return G.resize(siz), G;
    }
};
using namespace poly;

```

## 拉格朗日插值.cpp

```
#define DEBUG 1
#ifdef DEBUG
    #include "all.hpp"
#else
    #include <bits/stdc++.h>
#endif

using namespace std;
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
using db = long double;
using pdd = pair<db, db>;

const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

/*
记得先 init !!!

cal 函数传入一段连续的函数值(按照横坐标排序)
格式为 (x, f(x)), 然后传入待计算的横坐标

单次插值复杂度  $O(n)$  .
*/

namespace LagInt {
    ll fac[N];

    // 模数为质数 !!!
    ll qpow(ll base, ll k, ll mod) {
        if (base == 0) return 0;
        ll res = 1;
        base %= mod; base = (base + mod) % mod;
        k %= (mod - 1); k = (k + mod - 1) % (mod - 1);
        while (k) {
            if (k & 1) {
                res *= base; res %= mod;
            }
            k >>= 1;
            base *= base; base %= mod;
        }
        return res;
    }

    void init(int n) {
        fac[0] = 1;
    }
}
```

```

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

11 cal(vector<pll>& f, ll x) {
    int n = f.size() - 1;
    vector<ll> pre(n + 1, 0), suf(n + 1, 0);
    vector<ll> inv(n + 1, 0);

    for (int i = 0; i <= n; i++) {
        inv[i] = fac[i] * fac[n - i] % MOD;
    }
    for (int i = 0; i <= n; i++) {
        if (i - 1 >= 0) {
            pre[i] = pre[i - 1] * inv[i] % MOD;
        } else {
            pre[i] = inv[i];
        }
    }
    for (int i = n; i >= 0; i--) {
        if (i + 1 <= n) {
            suf[i] = suf[i + 1] * inv[i] % MOD;
        } else {
            suf[i] = inv[i];
        }
    }
    ll base = qpow(pre[n], MOD - 2, MOD);
    for (int i = 0; i <= n; i++) {
        inv[i] = base;
        if (i - 1 >= 0) {
            inv[i] = inv[i] * pre[i - 1] % MOD;
        }
        if (i + 1 <= n) {
            inv[i] = inv[i] * suf[i + 1] % MOD;
        }
    }

    for (int i = 0; i <= n; i++) {
        if (i - 1 >= 0) {
            pre[i] = pre[i - 1] * (x - f[i].first) % MOD;
            pre[i] = (pre[i] + MOD) % MOD;
        } else {
            pre[i] = (x - f[i].first) % MOD;
            pre[i] = (pre[i] + MOD) % MOD;
        }
    }

    for (int i = n; i >= 0; i--) {
        if (i + 1 <= n) {
            suf[i] = suf[i + 1] * (x - f[i].first) % MOD;
            suf[i] = (suf[i] + MOD) % MOD;
        } else {

```

```

        suf[i] = (x - f[i].first) % MOD;
        suf[i] = (suf[i] + MOD) % MOD;
    }
}

ll ans = 0;
for (int i = 0; i <= n; i++) {
    ll res = f[i].second;
    if (i - 1 >= 0) res = res * pre[i - 1] % MOD;
    if (i + 1 <= n) res = res * suf[i + 1] % MOD;
    res = res * inv[i] % MOD;
    if ((n - i) & 1) res = (-res + MOD) % MOD;
    ans = (ans + res) % MOD;
}
return ans;
}
};

```

## 拓展gcd.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  $X1 + k * dx$ ,  $Y1 + k * dy$ 
*/

template<typename T>
struct ExGcd{
    T a,b,dx,dy,c,gcd_ab;
    T X0,Y0,X1,Y1;
    bool init(T A,T B,T C = 0){
        a = A,b = B,c = C;
        gcd_ab = __gcd(a,b);
        if (c % gcd_ab != 0){
            return 0;
        }
        return 1;
    }
    void exgcd(T a,T 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;
}
};

```

## 数论分块.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;
    }
};

```

## 沃尔什卷积.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG 1
#define endl '\n'
#define FUCK if (DEBUG) cout << "fuck" << endl;
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 = 1e18;
const ll MOD = 1e9 + 7;

// XOR Fast Walsh-Hadamard Transform (in-place).
// 功能: 将向量 a 变换到 FWT 域并返回结果。
// 复杂度:  $O(n \log n)$ , 其中  $n = a.size()$ , 须为 2 的幂。
vector<ll> fwt(vector<ll>& a){
    int n = a.size();
    for(int len = 1; len < n; len <= 1)
        for(int i = 0; i < n; i += len << 1)
            for(int j = 0; j < len; ++j){
                ll u = a[i + j];
                ll v = a[i + j + len];
                a[i + j] = u + v;
                a[i + j + len] = u - v;
            }
    return a;
}

vector<ll> FWT(vector<ll>& a) {
    vector<ll> b = a;
    fwt(b);
    for (int i = 0; i < b.size(); i++) b[i] = b[i] * b[i];
    fwt(b);
    for (int i = 0; i < b.size(); i++) b[i] /= b.size();
    return b;
}

void solve() {
    ll n, m, k; cin >> n >> m >> k;
    ll M = (1 << m);
    vector<ll> a(M, 0);
    for (int i = 1; i < n + 1; i++) {
        string s; cin >> s;
        int x = 0;
        for (int j = 0; j < m; j++) {
            if (s[j] == 'A') {
                x += (1 << j);
            }
        }
    }
}
```



```

        a[x] ++;
    }

    a = FWT(a);

    for (int i = 0; i < M; i++) {
        for (int j = 0; j < m; j++) {
            if (i & (1 << j)) {
                a[i] += a[i ^ (1 << j)];
            }
        }
    }

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < M; j++) {
            if (j & (1 << i)) a[j] += a[j ^ (1 << i)];
        }
    }

    // for(int bit=0; bit<m; bit++)
    //     for(int mask=0; mask<M; mask++)
    //         if(mask & (1<<bit)) a[mask] += a[mask^(1<<bit)];

    ll ans = 0;
    for (int i = 1; i < M; i++) {
        int S = ((M - 1) ^ i);
        ll z = 1ll * n * n - a[S];
        if (z >= 2 * k) ans ++;
    }

    cout << ans << endl;
    return;
}

signed main() {
#ifdef 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();
    }

#ifdef 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;
}

```

## 矩阵.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;
}

```

## 类欧几里得算法.cpp

```

#include <bits/stdc++.h>
using namespace std;
using i128 = __int128_t;
using ll = long long;

// floor((a * i + b) / m)
// i 从 0 到 n - 1
i128 floor_sum_i128(i128 n, i128 m, i128 a, i128 b){
    if(n<=0) return 0;
    i128 ans=0;
    if(a<0){ b += a*(n-1); a = -a; }
    if(b<0){ i128 k = (-b + m - 1) / m; b += k*m; ans -= k*n; }
    while(true){
        if(a>=m){ ans += (a/m) * (n*(n-1)/2); a %= m; }
        if(b>=m){ ans += (b/m) * n; b %= m; }
        i128 y = a*n + b;
        if(y < m) break;
        n = y / m;
        b = y % m;
        i128 tmp = m; m = a; a = tmp;
    }
    return ans;
}

i128 floor_sum_mod(i128 n, i128 m, i128 a, i128 b, i128 mod){
    if(n<=0) return 0;
    a%=m; b%=m;
    i128 ans=0;
    if(a<0){ b += a*(n-1); a = -a; }

```

```

if(b<0){ i128 k = (-b + m - 1) / m; b += k*m; ans -= k*n; }
while(true){
    if(a>=m){ ans += (a/m) * (n*(n-1)/2); a %= m; }
    if(b>=m){ ans += (b/m) * n; b %= m; }
    i128 y = a*n + b;
    if(y < m) break;
    n = y / m;
    b = y % m;
    i128 tmp = m; m = a; a = tmp;
}
ans %= mod;
if(ans<0) ans += mod;
return ans;
}

```

## 线性基.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;
}
} ;

```

## 组合数.cpp

```

#include <bits/stdc++.h>
typedef long long ll;
const ll MOD = 1000000007;
using namespace std;
// 1e7 的数组慎开啊
// 支持广义二项式定理
namespace ComNum {
    const int MAXN = 5e6;
    ll fac[MAXN], inv[MAXN];

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

```

```

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

void init() {
    fac[0] = 1;
    for (int i = 1; i < MAXN; i++) {
        fac[i] = fac[i - 1] * i % MOD;
    }
    vector<ll> pre(MAXN, 1), suf(MAXN, 1);
    for (int i = 0; i < MAXN; i++) {
        if (i - 1 >= 0) pre[i] = pre[i - 1];
        pre[i] = pre[i] * fac[i] % MOD;
    }
    for (int i = MAXN - 1; i >= 0; i--) {
        if (i + 1 < MAXN) suf[i] = suf[i + 1];
        suf[i] = suf[i] * fac[i] % MOD;
    }
    ll base = qpow(pre.back(), MOD - 2, MOD);
    for (int i = 0; i < MAXN; i++) {
        ll fenzi = 1;
        if (i - 1 >= 0) fenzi = pre[i - 1] * fenzi % MOD;
        if (i + 1 < MAXN) fenzi = suf[i + 1] * fenzi % MOD;
        inv[i] = fenzi * base % MOD;
    }
    return;
}

ll C(ll n, ll m){
    if(m == 0) return 1;
    if(n >= 0){
        if(n < m) return 0;
        return fac[n] * inv[n - m] % MOD * inv[m] % MOD;
    }else{
        ll a = -n;
        ll nn = a + m - 1;
        ll res = fac[nn] * inv[nn - m] % MOD * inv[m] % MOD;
        if(m & 1) res = (MOD - res) % MOD;
        return res;
    }
}
};

```

## 自动取模.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 = 20000000;
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; }
};

```

## 质数检验.cpp

```

#include <bits/stdc++.h>
using namespace std;
using ull = unsigned long long;
using u128 = __uint128_t;

ull modmul(ull a, ull b, ull m){ return (u128)a*b % m; }
ull modpow(ull a, ull e, ull m){ u128 r=1, x=a%m; while(e){ if(e&1) r=(r*x)%m; x=(x*x)%m; e>>=1; } return (ull)r; }
bool miller_check(ull n, ull a){
    if(a % n == 0) return true;
    ull d = n-1; int s = 0;
    while((d&1)==0){ d >>= 1; ++s; }
    ull x = modpow(a, d, n);
    if(x==1 || x==n-1) return true;
    for(int i=1;i<s;i++){
        x = modmul(x, x, n);
        if(x==n-1) return true;
    }
    return false;
}

// 时间复杂度: O(k * log^3 n) (k为基础集合大小, 常数)
// 用法: if(isPrime(n)) // n <= 1e18

```

```

bool isPrime(ull n){
    if(n < 2) return false;
    for(ull p :
{2ull,3ull,5ull,7ull,11ull,13ull,17ull,19ull,23ull,29ull,31ull,37ull}){
        if(n == p) return true;
        if(n % p == 0) return false;
    }
    ull bases[] =
{2ull,325ull,9375ull,28178ull,450775ull,9780504ull,1795265022ull};
    for(ull a : bases) if(!miller_check(n, a)) return false;
    return true;
}

```

## 质数筛.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) {
    if (base == 0) return 0;
    ll res = 1;
    base %= mod; base = (base + mod) % mod;
    k %= (mod - 1); k = (k + mod - 1) % (mod - 1);
    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 {
                continue;
            }
        }
        if (ok) return i;
    }

    return 0;
}

```

## 逆序数.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);
    }
}

```

```
};
```

## 闵可夫斯基和.cpp

```
#define DEBUG 1
#define FUCK cout << "fuck" << endl;
#ifdef DEBUG
    #include "all.hpp"
#else
    #include <bits/stdc++.h>
#endif

using namespace std;
using ll = long long;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
using db = long double;
using pdd = pair<db, db>;
using i128 = __int128_t;

const ll N = 2000000;
const ll INF = 5e18;
const ll MOD = 1e9 + 7;

const db pi = acos(-1);
using T = ll;
struct P {
    T x, y;
    P() : x(0), y(0) {}
    P(T x, T y) : x(x), y(y) {}

    P operator+ (const P &o) { return P(x + o.x, y + o.y); }
    P operator- (const P &o) { return P(x - o.x, y - o.y); }
    P operator* (const T &o) { return P(x * o, y * o); }
    P operator/ (const T &o) { return P(x / o, y / o); }
    T operator* (const P &o) { return x * o.x + y * o.y; }
    T operator^ (const P &o) { return x * o.y - y * o.x; }

    bool operator== (const P &o) const {
        return x == o.x && y == o.y;
    }
    bool operator< (const P &o) const {
        return x < o.x || (x == o.x && y < o.y);
    }
}

T norm2() {
    return x * x + y * y;
}
db norm() {
    return sqrt(norm2());
}
```

```

    }
    P unit() {
        return *this / norm();
    }
    db angle() {
        db t = atan2(y, x);
        return t < 0 ? t + 2 * pi : t;
    }
    P rot(const T &o) {
        return P(x * cos(o) - y * sin(o), x * sin(o) + y * cos(o));
    }
};

vector<P> convex_hull(vector<P> pts) {
    int n = pts.size();
    if (n <= 1) return pts;

    sort(pts.begin(), pts.end());

    vector<P> lower, upper;
    for (auto &p : pts) {
        while (lower.size() >= 2 && ((lower.back() - lower[lower.size()-2]) ^ (p - lower.back())) <= 0)
            lower.pop_back();
        lower.push_back(p);
    }

    for (int i = n-1; i >= 0; i--) {
        P p = pts[i];
        while (upper.size() >= 2 && ((upper.back() - upper[upper.size()-2]) ^ (p - upper.back())) <= 0)
            upper.pop_back();
        upper.push_back(p);
    }

    lower.pop_back();
    upper.pop_back();
    lower.insert(lower.end(), upper.begin(), upper.end());
    return lower;
}

vector<P> minkowski(vector<P> P1, vector<P> P2) {
    int n = P1.size(), m = P2.size();

    rotate(P1.begin(), min_element(P1.begin(), P1.end()), P1.end());
    rotate(P2.begin(), min_element(P2.begin(), P2.end()), P2.end());

    vector<P> V1(n), V2(m);
    for (int i = 0; i < n; i++) {
        V1[i] = P1[(i + 1) % n] - P1[i];
    }
    for (int i = 0; i < m; i++) {
        V2[i] = P2[(i + 1) % m] - P2[i];
    }
}

```

```
vector<P> ans = {P1[0] + P2[0]};
int t = 0, i = 0, j = 0;
while (i < n && j < m) {
    int val = V1[i] ^ V2[j];
    if (val == 0) ans.emplace_back(ans.back() + V1[i++] + V2[j++]);
    else if (val > 0) ans.emplace_back(ans.back() + V1[i++]);
    else if (val < 0) ans.emplace_back(ans.back() + V2[j++]);
}
while (i < n) ans.emplace_back(ans.back() + V1[i++]);
while (j < m) ans.emplace_back(ans.back() + V2[j++]);
ans.pop_back();
return ans;
}
```

## 随机数.cpp

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

struct RandomNumberGenerator{
    RandomNumberGenerator() : gen(std::random_device{}()){
        ll generate(ll l, ll r){
            uniform_int_distribution<ll> dis(l, r);
            return dis(gen);
        }
        mt19937 gen;
    } gen;
}
```