# Standard Code Library

Your TeamName

Your School

October 3, 2023

# Contents

# 一切的开始

## 快读

```
#define gc()(is==it?it=(is=in)+fread(in,1,Q,stdin),(is==it?EOF:*is++):*is++)
const int Q=(1<<24)+1;
char in[Q],*is=in,*it=in,c;
void read(long long &n){
    for(n=0;(c=gc())<'0'||c>'9';);
    for(;c<='9'&&c>='0';c=gc())n=n*10+c-48;
}
```

## 代码模板

```
#include <bits/stdc++.h>

using namespace std;
#define dbg(x...) \
    do { \
        cout << #x << " -> "; \
        err(x); \
    } while (0)

void err() {
    cout << endl;
}

template<class T, class... Ts>
void err(T arg, Ts &... args) {
    cout << arg << ' ';
    err(args...);
}

typedef long long ll;
typedef pair<int, int> pii;
const int N = 1e5 + 10, Log = 20, inf = 0x3f3f3f3f;

void solve() {

}

int main() {
    int T = 1;
    ios::sync_with_stdio(false);
    cin >> T;
    while (T--) solve();
    return 0;
}
```

# 数据结构

## st 表

$st[i][j]$ 表示区间 $[i, i+2^j-1]$ 的 gcd

```
int st[N][Log + 5], logx[N];

void init(int n) {
    logx[0] = -1;
    for (int i = 1; i <= n; i++)logx[i] = logx[i >> 1] + 1;
    for (int i = 1; i <= n; i++)st[i][0] = i;
    for (int j = 1; (1 << j) <= n; j++) {
        for (int i = 1; i + (1 << j) - 1 <= n; i++) {
            st[i][j] = __gcd(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
        }
    }
}

int query(int l, int r){
```

```
    int k = logx[r - l + 1];
    return __gcd(st[l][k], st[r - (1 << k) + 1][k]);
}
```

## 树状数组

```
template <typename T>
struct Fenwick {
    const int n;
    vector<T> a;
    Fenwick(int n) : n(n), a(n + 1) {}
    void add(int x, T v) {
        while(x <= n){
            a[x] += v;
            x += x & -x;
        }
    }
    T sum(int x) {
        T ans = 0;
        for (int i = x; i; i -= i & -i) {
            ans += a[i];
        }
        return ans;
    }
    T rangeSum(int l, int r) {
        return sum(r) - sum(l - 1);
    }
};
```

## 主席树

```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
const int N = 200010;
int root[N], tot = 0, num[N], len, a[N];
struct Info {
    int sum, l, r;
} info[N << 5];

int getid(int x) {
    return lower_bound(num + 1, num + len + 1, x) - num;
}

void build(int &x, int l, int r) {//创建空树
    x = ++tot;
    info[x].sum = 0;
    if (l == r) return;
    int mid = (l + r) / 2;
    build(info[x].l, l, mid);
    build(info[x].r, mid + 1, r);
}

void update(int pre, int &now, int l, int r, int q) {//更新
    now = ++tot;
    info[now] = info[pre];
    info[now].sum++;
    if (l == r) return;
    int mid = (l + r) / 2;
    if (mid >= q) update(info[pre].l, info[now].l, l, mid, q);
    else update(info[pre].r, info[now].r, mid + 1, r, q);
}

int query(int pre, int now, int l, int r, int k) {//求第 k 小
    if (l == r) return l;
    int delta = info[info[now].l].sum - info[info[pre].l].sum;
    int mid = (l + r) / 2;
    if (delta >= k) return query(info[pre].l, info[now].l, l, mid, k);
```

```
    else return query(info[pre].r, info[now].r, mid + 1, r, k - delta);
}

int query_sum(int pre, int now, int l, int r, int k) {// 求小于等于 k 的个数
    if (l == r) return info[now].sum - info[pre].sum;
    int mid = (l + r) >> 1;
    if (k <= mid) return query_sum(info[pre].l, info[now].l, l, mid, k);
    else return (info[info[now].l].sum - info[info[pre].l].sum) + query_sum(info[pre].r, info[now].r, mid + 1, r, k);
}
/*
先进行离散化
sort(num + 1, num + 1 + n);
len = unique(num + 1, num + 1 + n) - num - 1;
建空树
build(root[0], 1, len);
更新
update(root[i - 1], root[i], 1, len, getid(a[i]));
查询 [l, r]
query_sum(root[l - 1], root[r], 1, len, k)
query(root[l - 1], root[r], 1, len, k)
*/
```

## 树链剖分

```
vector<int> e[N];

int n;
int sz[N], f[N], son[N];
int top[N], dfn[N], rk[N], tot, ru[N];
int a[N];

void dfs(int u, int fa){
    sz[u] = 1;
    f[u] = fa;
    son[u] = -1;
    for(int i : e[u]){
        if(i == fa) continue;
        dfs(i, u);
        sz[u] += sz[i];
        if(son[u] == -1 || sz[i] > sz[son[u]]) son[u] = i;
    }
}

void dfs1(int u, int t){
    top[u] = t;
    dfn[u] = ++tot;
    rk[tot] = u;
    if(son[u] == -1){
        ru[u] = dfn[u];
        return;
    }
    dfs1(son[u], t);
    for(int i : e[u]){
        if(i == f[u] || i == son[u]) continue;
        dfs1(i, i);
    }
    ru[u] = tot;
}

template<typename T>
struct SegmentTree{
    T sum[N << 2], lz[N << 2];
    void apply(int k, int l, int r, T x){
        sum[k] += (r - l + 1) * x;
        lz[k] += x;
    }
    void pd(int k, int l, int r){// push down
        int mid = (l + r) >> 1;
        apply(k << 1, l, mid, lz[k]);
        apply(k << 1 | 1, mid + 1, r, lz[k]);
        lz[k] = 0;
```

```
    }
    void pu(int k){// push up
        sum[k] = sum[k << 1] + sum[k << 1 | 1];
    }
    void build(int k, int l, int r){
        if(l == r){
            sum[k] = a[rk[l]];
            lz[k] = 0;
            return;
        }
        int mid = (l + r) >> 1;
        build(k << 1, l, mid);
        build(k << 1 | 1, mid + 1, r);
        pu(k);
    }
    void mdf(int k, int l, int r, int ql, int qr, T x){// modify [ql, qr] add x
        if(l > qr || r < ql) return;
        if(l >= ql && r <= qr){
            sum[k] += (r - l + 1) * x;
            lz[k] += x;
            return;
        }
        pd(k, l, r);
        int mid = (l + r) >> 1;
        mdf(k << 1, l, mid, ql, qr, x);
        mdf(k << 1 | 1, mid + 1, r, ql, qr, x);
        pu(k);
    }
    T query(int k, int l, int r, int ql, int qr){
        if(l > qr || r < ql) return 0;
        if(l >= ql && r <= qr){
            return sum[k];
        }
        pd(k, l, r);
        int mid = (l + r) >> 1;
        return query(k << 1, l, mid, ql, qr) + query(k << 1 | 1, mid + 1, r, ql, qr);
    }
};

SegmentTree<ll> seg;

int qrysum(int u, int v){
    int fu = top[u], fv = top[v], ret = 0;
    while(fu != fv){
        if(dfn[fu] > dfn[fv]){
            ret += seg.query(1, 1, n, dfn[fu], dfn[u]);
            u = f[fu];
        }else{
            ret += seg.query(1, 1, n, dfn[fv], dfn[v]);
            v = f[fv];
        }
        fu = top[u];
        fv = top[v];
    }
    if(dfn[u] > dfn[v]) swap(u, v);
    ret += seg.query(1, 1, n, dfn[u], dfn[v]);
    return ret;
}

void solve() {
    int m, rt;
    cin >> n;
    for(int i = 1; i <= n; i++) cin >> a[i];
    for(int i = 0, u, v; i < n - 1; i++){
        cin >> u >> v;
        e[u].push_back(v);
        e[v].push_back(u);
    }
    dfs(1, 0);
    dfs1(1, 1);
    seg.build(1, 1, n);
```

```
}
```

## 平衡树 Treap

### 普通平衡树

```
mt19937 mt(chrono::steady_clock::now().time_since_epoch().count());

int rng(int l, int r) {
    uniform_int_distribution<int> uni(l, r);
    return uni(mt);
}

struct Node{
    Node *lt, *rt; // 左右子结点
    int val, prio; // 值, 优先级
    int cnt, sz; // 重复次数, 子树大小

    Node(int _val) : val(_val), cnt(1), sz(1) {
        lt = rt = nullptr;
        prio = rng(1, 1e9);
    }

    void upd(){
        sz = cnt;
        if(lt != nullptr) sz += lt->sz;
        if(rt != nullptr) sz += rt->sz;
    }
};

struct Treap{
    int siz(Node *p){
        if(p == nullptr) return 0;
        return p->sz;
    }

    Node *root;

    pair<Node *, Node *> split(Node *cur, int key) { // 根据 val 分裂成 小于等于 key 和 大于 key 的两个 treap
        if (cur == nullptr) return {nullptr, nullptr};
        if (cur->val <= key) {// 当前属于第一个 treap
            auto temp = split(cur->rt, key);
            cur->rt = temp.first;
            cur->upd();
            return {cur, temp.second};
        } else {// 当前属于第二个 treap
            auto temp = split(cur->lt, key);
            cur->lt = temp.second;
            cur->upd();
            return {temp.first, cur};
        }
    }

    tuple<Node *, Node *, Node *> split_by_rk(Node *cur, int rk) { // 根据 rk 分裂成 小于 rk 和 等于 rk 和 大于 rk 的三个 treap, 其中第二个只有一个结点
        if (cur == nullptr) return {nullptr, nullptr, nullptr};
        int ls_siz = siz(cur->lt); // 左子树大小
        if (rk <= ls_siz) {// 当前属于第三个 treap
            Node *l, *mid, *r;
            tie(l, mid, r) = split_by_rk(cur->lt, rk);
            cur->lt = r;
            cur->upd();
            return {l, mid, cur};
        } else if (rk <= ls_siz + cur->cnt) {// 当前属于第二个 treap
            Node *lt = cur->lt;
            Node *rt = cur->rt;
            cur->lt = cur->rt = nullptr;
            return {lt, cur, rt};
        } else {// 当前属于第一个 treap
            Node *l, *mid, *r;
            tie(l, mid, r) = split_by_rk(cur->rt, rk - ls_siz - cur->cnt);
```

```
            cur->rt = l;
            cur->upd();
            return {cur, mid, r};
        }
    }

    Node *merge(Node *u, Node *v) {// 按照 prio 小根堆合并
        if (u == nullptr && v == nullptr) return nullptr;
        if (u != nullptr && v == nullptr) return u;
        if (v != nullptr && u == nullptr) return v;
        if (u->prio < v->prio) {
            u->rt = merge(u->rt, v);
            u->upd();
            return u;
        } else {
            v->lt = merge(u, v->lt);
            v->upd();
            return v;
        }
    }

    void insert(int val) {// 插入
        auto temp = split(root, val);
        auto l_tr = split(temp.first, val - 1);
        Node *new_node;
        if (l_tr.second == nullptr) {
            new_node = new Node(val);
        } else {
            l_tr.second->cnt++;
            l_tr.second->upd();
        }
        Node *l_tr_combined = merge(l_tr.first, l_tr.second == nullptr ? new_node : l_tr.second);
        root = merge(l_tr_combined, temp.second);
    }

    void del(int val) {// 删除
        auto temp = split(root, val);
        auto l_tr = split(temp.first, val - 1);
        if (l_tr.second == nullptr){
            root = merge(l_tr.first, temp.second);
            return;
        }
        if (l_tr.second->cnt > 1) {
            l_tr.second->cnt--;
            l_tr.second->upd();
            l_tr.first = merge(l_tr.first, l_tr.second);
        } else {
            if (temp.first == l_tr.second) {
                temp.first = nullptr;
            }
            delete l_tr.second;
            l_tr.second = nullptr;
        }
        root = merge(l_tr.first, temp.second);
    }

    int qrank_by_val(Node *cur, int val) { // 查询 val 的 rk
        auto temp = split(cur, val - 1);
        int ret = siz(temp.first) + 1;
        root = merge(temp.first, temp.second);
        return ret;
    }

    int qval_by_rank(Node *cur, int rk) { // 查询 rk 的 val 第 rk 大的值
        Node *l, *mid, *r;
        tie(l, mid, r) = split_by_rk(cur, rk);
        int ret = (mid == nullptr ? -114514 : mid->val);
        root = merge(merge(l, mid), r);
        return ret;
    }

```

```
    int qprev(int val) { // 查询第一个比 val 小的值
        auto temp = split(root, val - 1);
        int ret = qval_by_rank(temp.first, temp.first->sz);
        root = merge(temp.first, temp.second);
        return ret;
    }

    int qnex(int val) { // 查询第一个比 val 大的值
        auto temp = split(root, val);
        int ret = qval_by_rank(temp.second, 1);
        root = merge(temp.first, temp.second);
        return ret;
    }
};
```

### 区间翻转

```
mt19937 mt(chrono::steady_clock::now().time_since_epoch().count());

int rng(int l, int r) {
    uniform_int_distribution<int> uni(l, r);
    return uni(mt);
}

struct Node{
    Node *lt, *rt; // 左右子结点
    int val, prio; // 值, 优先级
    int cnt, sz; // 重复次数, 子树大小
    bool rev;// 是否翻转

    Node(int _val) : val(_val), cnt(1), sz(1) {
        lt = rt = nullptr;
        rev = false;
        prio = rng(1, 1e9);
    }

    void pu(){
        sz = cnt;
        if(lt != nullptr) sz += lt->sz;
        if(rt != nullptr) sz += rt->sz;
    }

    void pd(){
        if(rev){
            swap(lt, rt);
            if(lt != nullptr) lt->rev ^= 1;
            if(rt != nullptr) rt->rev ^= 1;
            rev = false;
        }
    }
};

struct Treap{
    Node* root;
    int siz(Node *p){
        if(p == nullptr) return 0;
        return p->sz;
    }

    pair<Node *, Node *> split(Node *cur, int sz){
        if(cur == nullptr) return {nullptr, nullptr};
        cur->pd();
        int lc = siz(cur->lt);
        if(sz <= lc){
            auto temp = split(cur->lt, sz);
            cur->lt = temp.second;
            cur->pu();
            return {temp.first, cur};
        }else{
            auto temp = split(cur->rt, sz - lc - cur->cnt);
            cur->rt = temp.first;
```

```
            cur->pu();
            return {cur,  temp.second};
        }
    }

    Node* merge(Node* u, Node* v) { // u 小 v 大
        if (u == nullptr && v == nullptr) return nullptr;
        if (u != nullptr && v == nullptr) return u;
        if (u == nullptr && v != nullptr) return v;
        u->pd(), v->pd();
        if (u->prio < v->prio) { // u 为根
            u->rt = merge(u->rt, v);
            u->pu();
            return u;
        } else {
            v->lt = merge(u, v->lt);
            v->pu();
            return v;
        }
    }

    void insert(int val){
        root = merge(root, new Node(val));
    }

    void seg_rev(int l, int r) {
        auto res = split(root, l - 1); // [1, l - 1] [l, n]
        auto ans = split(res.second, r - l + 1); // [l, r] [r + 1, n]
        ans.first->rev = true;
        root = merge(res.first, merge(ans.first, ans.second));
    }

    void print(Node* cur) {
        if (cur == nullptr) return;
        cur->pd();
        print(cur->lt);
        printf("%d ", cur->val);
        print(cur->rt);
    }
};
```

# 数学

## 组合数预处理

```
ll f[N], inv[N];

ll qpow(ll a, ll b) {
    ll res = 1;
    while (b) {
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b /= 2;
    }
    return res;
}

ll C(ll n, ll m) {
    return f[n] * inv[m] % mod * inv[n - m] % mod;
}

void init(int M) {
    f[0] = 1;
    for (int i = 1; i <= M; i++) f[i] = f[i - 1] * i % mod;
    inv[M] = qpow(f[M], mod - 2);
    for (int i = M - 1; i >= 0; i--) inv[i] = inv[i + 1] * (i + 1) % mod;
}
```

## Exgcd

求解 $xa+yb=c$

有解需满足 $gcd(a,b)|c$

设解出的一组特解为 $x_0,y_0$ 则通解为 $x=x_0+tb,y=y_0-ta$

```
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (!b) {
        x = 1;
        y = 0;
        return a;
    } else {
        ll g = exgcd(b, a % b, x, y);
        ll t = x;
        x = y;
        y = t - a / b * y;
        return g;
    }
}

ll upper(ll m, ll n) {//向上取整
    if (m <= 0) return m / n;
    return (m - 1) / n + 1;
}

ll lower(ll m, ll n) {//向下取整
    if (m >= 0) return m / n;
    return (m + 1) / n - 1;
}
```

## Lucas 定理

适用于模数为小质数

$$C_n^m \ mod \ p = C_{n \ mod \ p}^{m \ mod \ p} \times C_{\lfloor \frac{n}{p} \rfloor}^{\lfloor \frac{m}{p} \rfloor} \ mod \ p$$

```
ll C(ll n, ll r, ll p) {
    if (r > n || r < 0) return 0;
    return f[n] * inv[r] % p * inv[n - r] % p;
}

ll Lucas(ll n, ll m, ll p) {
    if (m == 0) return 1;
    return (C(n % p, m % p, p) * Lucas(n / p, m / p, p)) % p;
}
```

## 欧拉筛

```
const int N = 1e4 + 10, M = 10000;
vector<int> p;
int vis[N];

void init() {
    for (int i = 2; i <= M; i++) {
        if (!vis[i]) {
            p.push_back(i);
        }
        for (int j = 0; j < p.size() && p[j] * i <= M; j++) {
            vis[p[j] * i] = 1;
            if (i % p[j] == 0) {
                break;
            }
        }
    }
}
```

**求欧拉函数:** $phi(n)=n\prod(1-\frac{1}{p_i})$

```cpp
const int N = 1e4 + 10, M = 10000;
vector<int> p;
int phi[N], vis[N];

void rua() {//欧拉筛 以及 求欧拉函数
    for (int i = 2; i <= M; i++) {
        if (!vis[i]) {
            p.push_back(i);
            phi[i] = i - 1;
        }
        for (int j = 0; j < p.size() && p[j] * i <= M; j++) {
            vis[p[j] * i] = 1;
            if (i % p[j] == 0) {
                phi[i * p[j]] = phi[i] * p[j];
                break;
            } else {
                phi[i * p[j]] = phi[i] * phi[p[j]];
            }
        }
    }
}
```

## 线性基

线性基是一个数的集合，并且每个序列都拥有至少一个线性基，取线性基中若干个数异或起来可以得到原序列中的任何一个数。原序列里面的任意一个数都可以由线性基里面的一些数异或得到线性基里面的任意一些数异或起来都不能得到 0 线性基里面的数的个数唯一，并且在保持性质一的前提下，数的个数是最少的

```cpp
ll d[Log + 5];

void add(ll x){// 线性基插入
    for(int i = Log; i >= 0; i--){
        if((x >> i) & 1){
            if(d[i]) x ^= d[i];
            else{
                d[i] = x; // 插入成功
                break;
            }
        }
    }
}
```

## 欧拉降幂

$$a^b \pmod m \equiv a^{b \bmod \phi(m)+\phi(m)} \pmod m [b \geq \phi(m)]$$

**以下代码以计算 $a_l^{a_{l+1}^{a_{l+2}^{\cdots^{a_r}}}}$ 为例**

```cpp
unordered_map<ll, ll> mp;
ll a[N];
ll MOD(ll x, ll mod) {return x < mod ? x : x % mod + mod;}
ll qpow(ll a, ll b, ll mod) {
    ll res = 1;
    while (b) {
        if (b & 1) res = MOD(res * a, mod);
        b /= 2;
        a = MOD(a * a, mod);
    }
    return res;
}
ll phi(ll x) {
    if (mp[x]) return mp[x];
    ll res = x;
    for (ll i = 2; i * i <= x; i++) {
        if (x % i == 0) {
            res -= res / i;
            while (x % i == 0) x /= i;
        }
    }
```

```
    if (x > 1) {
        res -= res / x;
    }
    return mp[x] = res;
}
ll solve(int l, int r, ll p) {
    if (p == 1) return MOD(a[l], p);
    if (l == r) return MOD(a[l], p);

    return qpow(a[l], solve(l + 1, r, phi(p)), p);
}
```

### 矩阵快速幂

```
const int MOD = 1e9 + 7;

struct mat {
    int n;
    vector<vector<int>> a;

    mat(int n): n(n), a(n, vector<int>(n)){}

    mat operator*(const mat& b) const {
        mat res(n);
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                for (int k = 0; k < n; k++) {
                    (res.a[i][j] += 1ll * a[i][k] * b.a[k][j] % MOD) %= MOD;
                }
            }
        }
        return res;
    }

    void print(){
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                cout << a[i][j] << ' ';
            }
            cout << '\n';
        }
        cout << '\n';
    }
};

mat qpow(mat a, ll b) {
    mat res(a.n);
    for (int i = 0; i < a.n; i++) {
        res.a[i][i] = 1;
    }
    while (b) {
        if (b & 1) res = res * a;
        a = a * a, b >>= 1;
    }
    return res;
}
```

### 中国剩余定理

$x = num_i (mod\ r_i)$

```
ll CRT(int n) {//适用于 ri 两两互质
    ll N = 1, res = 0;
    for (int i = 1; i <= n; i++) N *= r[i];
    for (int i = 1; i <= n; i++) {
        ll m = N / r[i], x, y;
        exgcd(m, r[i], x, y);
        res = (res + num[i] * m % N * x % N) % N;
    }
```

```
    return (res + N) % N;
}
```

**通解解法**

$x = a_1(mod\ m_1)$

$x = a_2(mod\ m_2)$

$x = k_1 \times m_1 + a_1 = k_2 \times m_2 + a_2$

$k_1 \times m_1 - k_2 \times m_2 = a_2 - a_1$

**运用 exgcd 可求得一组解 (k1,k2) 可将上述两方程化为**

$x = k_1 \times m_1 + a_1(mod\ lcm(m_1, m_2))$

**若有多个方程依次两两合并即可**

## 整除分块

$\sum_{i=1}^{n} \lfloor \frac{n}{i} \rfloor$

```
ans = 0;
for(int l = 1, r; l <= n; l = r + 1) {
    r = n / (n / l);
    ans += n / l * (r - l + 1);
}
```

## 拉格朗日插值

设要求的 $n$ 次多项式为 $f(k)$, 已知 $f(x_i)\ (1 \le i \le n+1)$

$f(k) = \sum_{i=1}^{n+1} f(x_i) \prod_{j \ne i} \frac{k - x_j}{x_i - x_j}$

设要求的 $n$ 次多项式为 $f(k)$, 已知 $f(i)\ (1 \le i \le n+1)$

$f(k) = \sum_{i=1}^{n+1} f(i) \times \frac{\prod_{j=1}^{n+1}(x-j)}{(x-i)\times(-1)^{n+1-i}\times(i-1)!\times(n+1-i)!}$

以下代码求 $\sum_{i=1}^{n} i^k$

```
ll f[N], inv[N];

ll qpow(ll a, ll b) {
    ll res = 1;
    while (b) {
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b /= 2;
    }
    return res;
}

ll C(ll n, ll m) {
    return f[n] * inv[m] % mod * inv[n - m] % mod;
}

void init(int M) {
    f[0] = 1;
    for (int i = 1; i <= M; i++) f[i] = f[i - 1] * i % mod;
    inv[M] = qpow(f[M], mod - 2);
    for (int i = M - 1; i >= 0; i--) inv[i] = inv[i + 1] * (i + 1) % mod;
}

void solve() { // 对 k+1 次多项式插值, 且横坐标连续
    int n, k;
    cin >> n >> k;
```

```
    vector<ll> y(k + 3);
    for(int i = 1; i <= k + 2; i++){ // 前 k+2 项
        y[i] = (y[i - 1] + qpow(i, k)) % mod;
    }
    if(n <= k + 2){
        cout << y[n] << '\n';
        return;
    }
    init(2e6);
    vector<ll> p(k + 3);
    ll sum = 1;
    for(int i = 1; i <= k + 2; i++){
        p[i] = qpow(n - i, mod - 2);
        sum = sum * (n - i) % mod;
    }
    ll ans = 0;
    for(int i = 1; i <= k + 2; i++){
        ll tmp = y[i] * sum % mod * p[i] % mod * inv[i - 1] % mod * inv[k + 2 - i] % mod;
        if((k + 2 - i) & 1) ans -= tmp;
        else ans += tmp;
        ans %= mod;
        if(ans < 0) ans += mod;
    }
    cout << ans;
}
```

## FFT

```
typedef vector<int> vi;
typedef long long ll;
typedef pair<int, int> pii;

typedef complex<double> C;
typedef vector<double> vd;

void fft(vector<C> &a) {
    int n = (int) a.size(), L = 31 - __builtin_clz(n);
    static vector<complex<long double>> R(2, 1);
    static vector<C> rt(2, 1);
    for (static int k = 2; k < n; k *= 2) {
        R.resize(n);
        rt.resize(n);
        auto x = polar(1.0L, acos(-1.0L) / k);
        for (int i = k; i < 2 * k; i++) {
            rt[i] = R[i] = i & 1 ? R[i / 2] * x : R[i / 2];
        }
    }
    vi rev(n);
    for (int i = 0; i < n; i++) {
        rev[i] = (rev[i / 2] | (i & 1) << L) / 2;
    }
    for (int i = 0; i < n; i++) {
        if (i < rev[i]) swap(a[i], a[rev[i]]);
    }
    for (int k = 1; k < n; k *= 2) {
        for (int i = 0; i < n; i += 2 * k) {
            for (int j = 0; j < k; j++) {
                C z = rt[j + k] * a[i + j + k];
                a[i + j + k] = a[i + j] - z;
                a[i + j] += z;
            }
        }
    }
}

vd conv(const vd &a, const vd &b) {
    if (a.empty() || b.empty()) return {};
    vd res((int) a.size() + (int) b.size() - 1);
    int L = 32 - __builtin_clz((int) res.size()), n = 1 << L;
    vector<C> in(n), out(n);
    copy(a.begin(), a.end(), begin(in));
```

```
    for (int i = 0; i < (int) b.size(); i++) {
        in[i].imag(b[i]);
    }
    fft(in);
    for (C &x: in) x *= x;
    for (int i = 0; i < n; i++) {
        out[i] = in[-i & (n - 1)] - conj(in[i]);
    }
    fft(out);
    for (int i = 0; i < (int) res.size(); i++) {
        res[i] = imag(out[i]) / (4 * n);
    }
    return res;
}

using vll = vector<ll>;

vll gao(const vi &a, const vi &b) { //a 和 b 的卷积
    vd aa((int) a.size()), bb((int) b.size());
    for (int i = 0; i < (int) a.size(); i++) aa[i] = a[i];
    for (int j = 0; j < (int) b.size(); j++) bb[j] = b[j];

    vd cc = conv(aa, bb);
    vll c((int) cc.size());
    for (int i = 0; i < (int) c.size(); i++) c[i] = round(cc[i]);
    return c;
}
```

## NTT

```
typedef long long ll;
typedef __int128 i128;
const int N = 1e6 + 5;
const ll mod = 4179340454199820289, G = 3, Gi = 1393113484733273430;

ll qpow(ll a, ll b, ll p) {
    ll res = 1;
    while (b) {
        if (b & 1) res = (i128) res * a % p;
        b >>= 1;
        a = (i128) a * a % p;
    }
    return res;
}

ll f[N], g[N];
int bit, tot, rev[N];

void NTT(ll a[], int type) {
    for (int i = 0; i < tot; i++)
        if (i > rev[i])
            swap(a[i], a[rev[i]]);
    for (int mid = 1; mid < tot; mid <<= 1) {
        ll w1 = qpow(type == 1 ? G : Gi, (mod - 1) / (mid * 2), mod);
        for (int i = 0; i < tot; i += mid * 2) {
            ll wk = 1;
            for (int j = 0; j < mid; j++, wk = (i128) wk * w1 % mod) {
                ll x = a[i + j], y = (i128) wk * a[i + j + mid] % mod;
                a[i + j] = (x + y) % mod, a[i + j + mid] = (x - y + mod) % mod;
            }
        }
    }
    if (type == -1) {
        ll inv = qpow(tot, mod - 2, mod);
        for (int i = 0; i < tot; i++)
            a[i] = (i128) a[i] * inv % mod;
    }
}

void gao(){
    int n = 0, m = 0;// f, g 长度
```

```
    for(int i = 0; i < n; i++) f[i] = 0;
    for (int i = 0; i < m; i++) g[i] = 0;
    while ((1 << bit) <= n + m) bit++;
    tot = 1 << bit;
    for (int i = 0; i < tot; i++)
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
    for (int i = n; i < tot; i++) f[i] = 0;
    for (int i = m; i < tot; i++) g[i] = 0;
    NTT(f, 1), NTT(g, 1);
    for (int i = 0; i < tot; i++) f[i] = (i128) f[i] * g[i] % mod;
    NTT(f, -1);
}
```

## 高斯消元

```
const double eps = 1e-8;

int sgn(double x) {
    if (fabs(x) < eps) return 0;
    if (x < 0) return -1;
    return 1;
}

double a[4][4], x[4], b[4][4], y[4];
int equ, var;

int Gauss() {
    int i, j, k, col, max_r;
    for (k = 0, col = 0; k < equ && col < var; ++k, ++col) {
        max_r = k;
        for (i = k + 1; i < equ; ++i) if (fabs(a[i][col]) > fabs(a[max_r][col])) max_r = i;
        if (fabs(a[max_r][col]) < eps) return 0;
        if (k != max_r) {
            for (j = col; j < var; ++j) swap(a[k][j], a[max_r][j]);
            swap(x[k], x[max_r]);
        }
        x[k] /= a[k][col];
        for (j = col + 1; j < var; ++j) a[k][j] /= a[k][col];
        a[k][col] = 1;
        for (i = 0; i < equ; ++i) {
            if (i != k) {
                x[i] -= x[k] * a[i][col];
                for (j = col + 1; j < var; ++j) a[i][j] -= a[k][j] * a[i][col];
                a[i][col] = 0;
            }
        }
    }
    return 1;
}

int Gauss(int n, int m){// equ var
    equ = n;
    var = m;
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            a[i][j] = b[i][j];
        }
        x[i] = y[i];
    }
    if(!Gauss()) return 0;
    for(int i = 0; i < n; i++){
        double res = 0;
        for(int j = 0; j < m; j++){
            res += x[j] * b[i][j];
        }
        if(sgn(res - y[i])) return 0;
    }
    return 1;
}
```

# 图论

## LCA

### 倍增求法

```cpp
const int N = 100010, Log = 20;
int anc[N][Log + 5], depth[N];
vector<int> e[N];

void dfs(int u, int fa) {
    anc[u][0] = fa;
    depth[u] = depth[fa] + 1;
    for (int i : e[u]) {
        if(i == fa) continue;
        dfs(i, u);
    }
}

void init(int root, int n) {//初始化
    depth[0] = 0;
    dfs(root, 0);
    for (int j = 1; j <= Log; j++) {
        for (int i = 1; i <= n; i++) {
            anc[i][j] = anc[anc[i][j - 1]][j - 1];
        }
    }
}

int rush(int u, int h) {
    for (int i = 0; i <= Log; i++) {
        if (h >> i & 1) u = anc[u][i];
    }
    return u;
}

int qry(int x, int y) {// 求 x 和 y 的 lca
    if (depth[x] < depth[y]) swap(x, y);
    x = rush(x, depth[x] - depth[y]);
    if (x == y) return x;
    for (int i = Log; i >= 0; i--) {
        if (anc[x][i] != anc[y][i]) {
            x = anc[x][i];
            y = anc[y][i];
        }
    }
    return anc[x][0];
}
```

# 计算几何

## 二维几何

```cpp
namespace Geometry {
using T = ll;
constexpr T eps = 0;

bool eq(const T &x, const T &y) { return abs(x - y) <= eps; }
inline constexpr int type(T x, T y) {
    if(x == 0 and y == 0) return 0;
    if(y < 0 or (y == 0 and x > 0)) return -1;
    return 1;
}
struct Point {
    T x, y;
    constexpr Point(T _x = 0, T _y = 0) : x(_x), y(_y) {}
    constexpr Point operator+() const noexcept { return *this; }
    constexpr Point operator-() const noexcept { return Point(-x, -y); }
    constexpr Point operator+(const Point &p) const { return Point(x + p.x, y + p.y); }
```

```
    constexpr Point operator-(const Point &p) const { return Point(x - p.x, y - p.y); }
    constexpr Point &operator+=(const Point &p) { return x += p.x, y += p.y, *this; }
    constexpr Point &operator-=(const Point &p) { return x -= p.x, y -= p.y, *this; }
    constexpr T operator*(const Point &p) const { return x * p.x + y * p.y; }
    constexpr Point &operator*=(const T &k) { return x *= k, y *= k, *this; }
    constexpr Point operator*(const T &k) { return Point(x * k, y * k); }
    constexpr bool operator==(const Point &r) const noexcept { return r.x == x and r.y == y; }
    constexpr T cross(const Point &r) const { return x * r.y - y * r.x; }

    constexpr bool operator<(const Point &r) const { return pair(x, y) < pair(r.x, r.y); }

    // 1 : left, 0 : same, -1 : right
    constexpr int toleft(const Point &r) const {
        auto t = cross(r);
        return t > eps ? 1 : t < -eps ? -1 : 0;
    }

    constexpr bool arg_cmp(const Point &r) const {
        int L = type(x, y), R = type(r.x, r.y);
        if(L != R) return L < R;

        T X = x * r.y, Y = r.x * y;
        if(X != Y) return X > Y;
        return x < r.x;
    }
};
bool arg_cmp(const Point &l, const Point &r) { return l.arg_cmp(r); }
ostream &operator<<(ostream &os, const Point &p) { return os << p.x << " " << p.y; }
istream &operator>>(istream &is, Point &p) {
    is >> p.x >> p.y;
    return is;
}

struct Line {
    Point a, b;
    Line() = default;
    Line(Point a, Point b) : a(a), b(b) {}
    // ax + by = c
    Line(T A, T B, T C) {
        if(A == 0) {
            a = Point(0, C / B), b = Point(1, C / B);
        } else if(B == 0) {
            a = Point(C / A, 0), b = Point(C / A, 1);
        } else {
            a = Point(0, C / B), b = Point(C / A, 0);
        }
    }
    // 1 : left, 0 : same, -1 : right
    constexpr int toleft(const Point &r) const {
        auto t = (b - a).cross(r - a);
        return t > eps ? 1 : t < -eps ? -1 : 0;
    }

    friend std::ostream &operator<<(std::ostream &os, Line &ls) {
        return os << "{"
                  << "(" << ls.a.x << ", " << ls.a.y << "), (" << ls.b.x << ", " << ls.b.y << ")}";
    }
};
istream &operator>>(istream &is, Line &p) { return is >> p.a >> p.b; }

struct Segment : Line {
    Segment() = default;
    Segment(Point a, Point b) : Line(a, b) {}
};

ostream &operator<<(ostream &os, Segment &p) { return os << p.a << " to " << p.b; }
istream &operator>>(istream &is, Segment &p) {
    is >> p.a >> p.b;
    return is;
}

```

```cpp
struct Circle {
    Point p;
    T r;
    Circle() = default;
    Circle(Point p, T r) : p(p), r(r) {}
};

using pt = Point;
using Points = vector<pt>;
using Polygon = Points;
T cross(const pt &x, const pt &y) { return x.x * y.y - x.y * y.x; }
T dot(const pt &x, const pt &y) { return x.x * y.x + x.y * y.y; }

T abs2(const pt &x) { return dot(x, x); }
// http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=CGL_1_C
// 点の回転方向
int ccw(const Point &a, Point b, Point c) {
    b = b - a, c = c - a;
    if(cross(b, c) > 0) return +1;   // "COUNTER_CLOCKWISE"
    if(cross(b, c) < 0) return -1;   // "CLOCKWISE"
    if(dot(b, c) < 0) return +2;     // "ONLINE_BACK"
    if(abs2(b) < abs2(c)) return -2; // "ONLINE_FRONT"
    return 0;                        // "ON_SEGMENT"
}

// http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=CGL_2_A
// 平行判定
bool parallel(const Line &a, const Line &b) { return (cross(a.b - a.a, b.b - b.a) == 0); }

// http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=CGL_2_A
// 垂直判定
bool orthogonal(const Line &a, const Line &b) { return (dot(a.a - a.b, b.a - b.b) == 0); }

bool intersect(const Line &l, const Point &p) { return abs(ccw(l.a, l.b, p)) != 1; }

bool intersect(const Line &l, const Line &m) { return !parallel(l, m); }

bool intersect(const Segment &s, const Point &p) { return ccw(s.a, s.b, p) == 0; }

bool intersect(const Line &l, const Segment &s) { return cross(l.b - l.a, s.a - l.a) * cross(l.b - l.a, s.b - l.a) <=
 ↪  0; }

// http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=CGL_2_B
bool intersect(const Segment &s, const Segment &t) { return ccw(s.a, s.b, t.a) * ccw(s.a, s.b, t.b) <= 0 && ccw(t.a,
 ↪  t.b, s.a) * ccw(t.a, t.b, s.b) <= 0; }

bool intersect(const Polygon &ps, const Polygon &qs) {
    int pl = si(ps), ql = si(qs), i = 0, j = 0;
    while((i < pl or j < ql) and (i < 2 * pl) and (j < 2 * ql)) {
        auto ps0 = ps[(i + pl - 1) % pl], ps1 = ps[i % pl];
        auto qs0 = qs[(j + ql - 1) % ql], qs1 = qs[j % ql];
        if(intersect(Segment(ps0, ps1), Segment(qs0, qs1))) return true;
        Point a = ps1 - ps0;
        Point b = qs1 - qs0;
        T v = cross(a, b);
        T va = cross(qs1 - qs0, ps1 - qs0);
        T vb = cross(ps1 - ps0, qs1 - ps0);

        if(!v and va < 0 and vb < 0) return false;
        if(!v and !va and !vb) {
            i += 1;
        } else if(v >= 0) {
            if(vb > 0)
                i += 1;
            else
                j += 1;
        } else {
            if(va > 0)
                j += 1;
            else
                i += 1;
```

```
        }
    }
    return false;
}

T norm(const Point &p) { return p.x * p.x + p.y * p.y; }
Point projection(const Segment &l, const Point &p) {
    T t = dot(p - l.a, l.a - l.b) / norm(l.a - l.b);
    return l.a + (l.a - l.b) * t;
}

Point crosspoint(const Line &l, const Line &m) {
    T A = cross(l.b - l.a, m.b - m.a);
    T B = cross(l.b - l.a, l.b - m.a);
    if(A == 0 and B == 0) return m.a;
    return m.a + (m.b - m.a) * (B / A);
}

// http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=CGL_2_C
Point crosspoint(const Segment &l, const Segment &m) { return crosspoint(Line(l), Line(m)); }

// http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=CGL_3_B
// 凸性判定
bool is_convex(const Points &p) {
    int n = (int)p.size();
    for(int i = 0; i < n; i++) {
        if(ccw(p[(i + n - 1) % n], p[i], p[(i + 1) % n]) == -1) return false;
    }
    return true;
}

Points convex_hull(Points p) {
    int n = p.size(), k = 0;
    if(n <= 2) return p;
    sort(begin(p), end(p), [](pt x, pt y) { return (x.x != y.x ? x.x < y.x : x.y < y.y); });
    Points ch(2 * n);
    for(int i = 0; i < n; ch[k++] = p[i++]) {
        while(k >= 2 && cross(ch[k - 1] - ch[k - 2], p[i] - ch[k - 1]) <= 0) --k;
    }
    for(int i = n - 2, t = k + 1; i >= 0; ch[k++] = p[i--]) {
        while(k >= t && cross(ch[k - 1] - ch[k - 2], p[i] - ch[k - 1]) <= 0) --k;
    }
    ch.resize(k - 1);
    return ch;
}

// 面積の 2 倍
T area2(const Points &p) {
    T res = 0;
    rep(i, si(p)) { res += cross(p[i], p[i == si(p) - 1 ? 0 : i + 1]); }
    return res;
}

enum { _OUT, _ON, _IN };

int contains(const Polygon &Q, const Point &p) {
    bool in = false;
    for(int i = 0; i < Q.size(); i++) {
        Point a = Q[i] - p, b = Q[(i + 1) % Q.size()] - p;
        if(a.y > b.y) swap(a, b);
        if(a.y <= 0 && 0 < b.y && cross(a, b) < 0) in = !in;
        if(cross(a, b) == 0 && dot(a, b) <= 0) return _ON;
    }
    return in ? _IN : _OUT;
}

Polygon Minkowski_sum(const Polygon &P, const Polygon &Q) {
    vector<Segment> e1(P.size()), e2(Q.size()), ed(P.size() + Q.size());
    const auto cmp = [](const Segment &u, const Segment &v) { return (u.b - u.a).arg_cmp(v.b - v.a); };
    rep(i, P.size()) e1[i] = {P[i], P[(i + 1) % P.size()]};
    rep(i, Q.size()) e2[i] = {Q[i], Q[(i + 1) % Q.size()]};
```

```
    rotate(begin(e1), min_element(all(e1), cmp), end(e1));
    rotate(begin(e2), min_element(all(e2), cmp), end(e2));
    merge(all(e1), all(e2), begin(ed), cmp);
    const auto check = [](const Points &res, const Point &u) {
        const auto back1 = res.back(), back2 = *prev(end(res), 2);
        return eq(cross(back1 - back2, u - back2), eps) and dot(back1 - back2, u - back1) >= -eps;
    };
    auto u = e1[0].a + e2[0].a;
    Points res{u};
    res.reserve(P.size() + Q.size());
    for(const auto &v : ed) {
        u = u + v.b - v.a;
        while(si(res) >= 2 and check(res, u)) res.pop_back();
        res.eb(u);
    }
    if(res.size() and check(res, res[0])) res.pop_back();
    return res;
}

// -1 : on, 0 : out, 1 : in
// O(log(n))
int is_in(const Polygon &p, const Point &a) {
    if(p.size() == 1) return a == p[0] ? -1 : 0;
    if(p.size() == 2) return intersect(Segment(p[0], p[1]), a);
    if(a == p[0]) return -1;
    if((p[1] - p[0]).toleft(a - p[0]) == -1 || (p.back() - p[0]).toleft(a - p[0]) == 1) return 0;
    const auto cmp = [&](const Point &u, const Point &v) { return (u - p[0]).toleft(v - p[0]) == 1; };
    const size_t i = lower_bound(p.begin() + 1, p.end(), a, cmp) - p.begin();
    if(i == 1) return intersect(Segment(p[0], p[i]), a) ? -1 : 0;
    if(i == p.size() - 1 && intersect(Segment(p[0], p[i]), a)) return -1;
    if(intersect(Segment(p[i - 1], p[i]), a)) return -1;
    return (p[i] - p[i - 1]).toleft(a - p[i - 1]) > 0;
}

Points halfplane_intersection(vector<Line> L, const T inf = 1e9) {
    Point box[4] = {Point(inf, inf), Point(-inf, inf), Point(-inf, -inf), Point(inf, -inf)};
    rep(i, 4) { L.emplace_back(box[i], box[(i + 1) % 4]); }
    sort(all(L), [](const Line &l, const Line &r) { return (l.b - l.a).arg_cmp(r.b - r.a); });
    deque<Line> dq;
    int len = 0;
    auto check = [](const Line &a, const Line &b, const Line &c) { return a.toleft(crosspoint(b, c)) == -1; };
    rep(i, L.size()) {
        while(dq.size() > 1 and check(L[i], *(end(dq) - 2), *(end(dq) - 1))) dq.pop_back();
        while(dq.size() > 1 and check(L[i], dq[0], dq[1])) dq.pop_front();
        // dump(L[i], si(dq));

        if(dq.size() and eq(cross(L[i].b - L[i].a, dq.back().b - dq.back().a), 0)) {
            if(dot(L[i].b - L[i].a, dq.back().b - dq.back().a) < eps) return {};
            if(L[i].toleft(dq.back().a) == -1)
                dq.pop_back();
            else
                continue;
        }
        dq.emplace_back(L[i]);
    }

    while(dq.size() > 2 and check(dq[0], *(end(dq) - 2), *(end(dq) - 1))) dq.pop_back();
    while(dq.size() > 2 and check(dq.back(), dq[0], dq[1])) dq.pop_front();
    if(si(dq) < 3) return {};
    Polygon ret(dq.size());
    rep(i, dq.size()) ret[i] = crosspoint(dq[i], dq[(i + 1) % dq.size()]);
    return ret;
}
} // namespace Geometry

using namespace Geometry;
```

## Andrew

```
const double eps = 1e-9, pi = acos(-1.0);
const int N = 1e5 + 10;
```

```

int n, cnt, m;

int sgn(double x) {
    if(fabs(x) < eps)   return 0;
    if(x > 0)   return 1;
    return -1;
}

struct point {
    double x, y;
    point(double a = 0.0, double b = 0.0) : x(a), y(b) {}
    bool operator < (point t) {
        if(sgn(x - t.x) == 0)  return y < t.y;
        return x < t.x;
    }
    point operator - (point p){
        return {x - p.x, y - p.y};
    }
    double operator ^ (point p){
        return x * p.y - y * p.x;
    }
}p[N], ans[N];

double dis(point a, point b) {
    a = a - b;
    return sqrt(a.x * a.x + a.y * a.y);
}

void Andrew() {
    sort(p, p + n);
    int p1 = 0, p2;
    for(int i = 0; i < n; i++) {
        while(p1 > 1 && sgn((ans[p1] - ans[p1 - 1]) ^ (p[i] - ans[p1 - 1])) <= 0)   p1--;
        ans[++p1] = p[i];
    }
    p2 = p1;
    for(int i = n - 2; i >= 0; i--) {
        while(p2 > p1 && sgn((ans[p2] - ans[p2 - 1]) ^ (p[i] - ans[p2 - 1])) <= 0)  p2--;
        ans[++p2] = p[i];
    }
    double target = 0.0;
    for(int i = 1; i < p2; i++){
        target += dis(ans[i], ans[i + 1]);
    }
    printf("%.2f\n", target);
}
/*
usage:
scanf("%d", &n);
for(int i = 0; i < n; i++)
    scanf("%lf%lf", &p[i].x, &p[i].y);
Andrew();
*/
```

## CHT

```
// 维护上凸壳
struct Line {
    ll k, b;
    double intersect(Line l) {
        //交点 x 坐标
        double db = l.b - b;
        double dk = k - l.k;
        return db / dk;
    }

    ll calc (int x) {
        return k * x + b;
    }
};
```

```
struct CHT {
    vector<double> x; // 相邻线交点
    vector<Line> line; // 线

    void init(Line l) {
        x.push_back(-inf);
        line.push_back(l);
    }

    void addLine(Line l) {
        while (line.size() >= 2 && l.intersect(line[line.size() - 2]) <= x.back()) {
            x.pop_back();
            line.pop_back();
        }
        x.push_back(l.intersect(line.back()));
        line.push_back(l);
    }

    ll query(int qx) {
        int id = upper_bound(x.begin(), x.end(), qx) - x.begin() - 1; // 计算点属于的线 id
        return line[id].calc(qx);
    }
};
```

# 字符串

## KMP

```
int nxt[N];
string a, b;
//a 为模式串  b 为匹配串

int kmp(int n, int m){
    int res = 0;
    nxt[0] = -1;
    for(int j = -1, i = 0; i < n;){
        if(j == -1 || a[j] == a[i]){
            i++;j++;
            nxt[i] = j;
        }else{
            j = nxt[j];
        }
    }
    //i 模式串  j 匹配串
    for(int i = 0, j = 0; j < m; ){
        if(i == -1 || a[i] == b[j]){
            i++;j++;
        }else i = nxt[i];
        if(i == n){
            res += 1;
            // position:: j - n + 1
            i = nxt[i];
        }
    }
    return res;
}
```

## 序列自动机

```
构建:
for(int i = n; i >= 1; i--){
    for(int j = 0; j < 26; j++) ne[i - 1][j] = ne[i][j];
    ne[i - 1][s[i - 1] - 'a'] = i;
}

求三 (或多个) 个串的公共子序列个数:
int dfs(int p1, int p2, int p3){
    if(f[p1][p2][p3]) return f[p1][p2][p3];
```

```
    for(int i = 0; i < 26; i++){
        if(ne[0][p1][i] && ne[1][p2][i] && ne[2][p3][i]){
            f[p1][p2][p3] = (f[p1][p2][p3] + dfs(ne[0][p1][i], ne[1][p2][i], ne[2][p3][i])) % mod;
        }
    }
    f[p1][p2][p3] = (f[p1][p2][p3] + 1) % mod;
    return f[p1][p2][p3];
}
```

## 字典树

```
//对数排序 查找排序后第 k 个数 每个数 <= 1e9
const int N = 5000010;// 总长度
int trie[N][10], tot, sum[N][10], ssum = 0;
int color[N];

void insert(int x){
    int r = 1e9, p = 0;
    ssum++;
    for(int i = 0; i < 10; i++, r /= 10){
        int c = x / r;
        c %= 10;
        sum[p][c]++;
        if(!trie[p][c]) trie[p][c] = ++tot;
        p = trie[p][c];
    }
    color[p]++;
}

int find(int k){
    int res = 0, p = 0;
    while(k > 0){
        for(int i = 0; i < 10; i++){
            if(sum[p][i] < k) k -= sum[p][i];
            else{
                res = res * 10 + i;
                p = trie[p][i];
                k -= color[p];
                break;
            }
        }
    }
    return res;
}
```

## 字符串双哈希

```
#include <bits/stdc++.h>

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

const int N = 1e5 + 10;
const pii mod = {1e9 + 7, 1e9 + 9};
const pii base = {131, 251};
pll pw[N];

pll operator*(const pll &p1, const pll &p2) {
    return {p1.first * p2.first % mod.first, p1.second * p2.second % mod.second};
}

pll operator+(const pll &p1, const pll &p2) {
    return {(p1.first + p2.first) % mod.first, (p1.second + p2.second) % mod.second};
}

pll operator-(const pll &p1, const pll &p2) {
    return {(p1.first - p2.first + mod.first) % mod.first, (p1.second - p2.second + mod.second) % mod.second};
}
```

```

struct Hash {
    vector<pll> f;
    int n{};

    void init(ll ss[], int _n) {
        n = _n;
        f.resize(n + 1, {0, 0});
        for (int i = 1; i <= n; i++) {
            ll ch = ss[i];
            f[i] = f[i - 1] * base + pll{ch, ch};
        }
    }

    pll ask(int l, int r) {//[l, r]
        return f[r] - f[l - 1] * pw[r - l + 1];
    }
};
//记得初始化 pw
//pw[0] = {1, 1};
//for (int i = 1; i <= n; i++) pw[i] = pw[i - 1] * base;
```

# 杂项

## 莫队

时间复杂度 $O(\frac{n^2}{S}+mS)$, $n$ 为长度, $m$ 个询问, 块长为 $S$ (一般取 $\sqrt{n}$ 或 $\frac{n}{\sqrt{m}}$)

```
int unit;
int a[N];

struct node {
    int l, r, id;

    bool operator < (const node &k) const {
        if (l / unit != k.l / unit) return l / unit < k.l / unit;
        return r < k.r;
    }
} q[N];
void add(int i) {

}

void sub(int i) {

}
void solve(){
    unit = (int)sqrt(m);// m 个区间
    sort(q + 1, q + 1 + m);
    int L = 1, R = 0;
    for (int i = 1; i <= m; i++) {
        while (R < q[i].r) {
            R++;
            add(R);
        }
        while (R > q[i].r) {
            sub(R);
            R--;
        }
        while (L > q[i].l) {
            L--;
            add(L);
        }
        while (L < q[i].l) {
            sub(L);
            L++;
        }
    }
}
```

## unordered_map

```cpp
struct HashFunc{
    static uint64_t splitmix64(uint64_t x) {
        // http://xorshift.di.unimi.it/splitmix64.c
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }
    template<typename T, typename U>
    size_t operator()(const std::pair<T, U>& p) const {
        static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().time_since_epoch().count();
        return splitmix64(p.first + FIXED_RANDOM) ^ splitmix64(p.second + FIXED_RANDOM);
    }
};

// 键值比较，哈希碰撞的比较定义，需要直到两个自定义对象是否相等
struct EqualKey {
    template<typename T, typename U>
    bool operator ()(const std::pair<T, U>& p1, const std::pair<T, U>& p2) const {
        return p1.first == p2.first && p1.second == p2.second;
    }
};
unordered_map<pii, int, HashFunc, EqualKey> mp;
```

## DSU

```cpp
struct DSU{
    int f[N];
    void init(int n){
        for(int i = 0; i <= n; i++) f[i] = -1;
    }
    int find(int x){
        return f[x] < 0 ? x : f[x] = find(f[x]);
    }
    void merge(int x, int y){
        int fx = find(x), fy = find(y);
        if(fx == fy) return;
        if(f[fx] > f[fy]) swap(fx, fy);
        f[fx] += f[fy];
        f[fy] = fx;
    }
}dsu;
```

## Floyd 判圈

```cpp
//适用于每个点出度唯一的图找环
const ll mod = 1099511627776;

ll calc(ll x){
    return (x + (x >> 20) + 12345) % mod;
}

void Floyd_Cycle_Detection_Algorithm(){
    ll p1 = 1611516670, p2 = 1611516670; // 起始点
    do{
        p1 = calc(p1); // 移动一次
        p2 = calc(calc(p2)); // 移动两次
    }while(p1 != p2);
    // 存在环
    ll len = 0;// 环长
    do{
        p2 = calc(p2);
        len++;
    }while(p1 != p2);
    p1 = 1611516670;// 寻找环起点
    ll c1 = 0; // 起点到环起点的距离
    while(p1 != p2){
        p1 = calc(p1);
```

```
        p2 = calc(p2);
        c1++;
    }
    cout << p1 << ' ' << len << ' ' << c1 << '\n';
}
```

### 三分搜索

```
auto ternary_search = [&](ld l, ld r) {
    int it = 300;              //set the error limit here
    while (it--) {
        ld m1 = l + (r - l) / 3;
        ld m2 = r - (r - l) / 3;
        ld f1 = f(m1);      //evaluates the function at m1
        ld f2 = f(m2);      //evaluates the function at m2
        if (f1 < f2)
            l = m1;
        else
            r = m2;
    }
    return l;                    //return the maximum of f(x) in [l, r]
};
```

```
auto ternary_search = [&](int l, int r) {
    int it = 300;              //set the error limit here
    while (it--) {
        int m1 = (l + r) >> 1;
        int m2 = m1 + 1;
        ld f1 = f(m1);      //evaluates the function at m1
        ld f2 = f(m2);      //evaluates the function at m2
        if (f1 < f2)
            l = m1;
        else
            r = m2;
    }
    return l;                    //return the maximum of f(x) in [l, r]
};
```

### 随机器

```
// random shuffle
random_device rd;
mt19937 rng(rd());
shuffle(a + 1, a + 1 + n, rng);
```

```
// 区间随机
mt19937 mt(chrono::steady_clock::now().time_since_epoch().count());

int rng(int l, int r) {
    uniform_int_distribution<int> uni(l, r);
    return uni(mt);
}
```