unordered_map玄学优化

```
#include <chrono>
#include <unordered_map>
using namespace std;

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

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().\
            time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};

unordered_map<long long, int, custom_hash> safe_map;
```

st-lca

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

const int maxn = 1e5 + 5;

vector<int> G[maxn];
int ver[maxn << 1], tot, First[maxn], dep[maxn << 1];
int dp[maxn << 1][20];
bool vis[maxn];

inline void init() {
    tot = 0;
    memset(vis, false, sizeof vis);
}

inline void addedge(int u, int v) {
    G[u].push_back(v);
    G[v].push_back(u);
}

void DFS(int u, int d) {
    vis[u] = true, ver[++tot] = u, First[u] = tot, dep[tot] = d;
    for (const int &v : G[u]) {
        if (!vis[v]) {
            DFS(v, d + 1);
```

```
            ver[++tot] = u, dep[tot] = d;
        }
    }
}

inline void ST() {
    for (int i = 1; i <= tot; i++) dp[i][0] = i;
    for (int j = 1; (1 << j) - 1<= tot; j++) {
        for (int i = 1; i + (1 << j) - 1 <= tot; i++) {
            int x = dp[i][j - 1], y = dp[i + (1 << (j - 1))][j - 1];
            dp[i][j] = (dep[x] < dep[y]) ? x : y;
        }
    }
}

inline int rmq(int l, int r) {
    int k = 0;
    while ((1 << (k + 1)) <= r - l + 1) ++k;
    int x = dp[l][k], y = dp[r - (1 << k) + 1][k];
    return dep[x] < dep[y] ? x : y;
}

inline int lca(int u, int v) {
    int x = First[u], y = First[v];
    if (x > y) swap(x, y);
    return ver[rmq(x, y)];
}

// inline int lca(int u, int v) {
//     int l = First[u], r = First[v];
//     if (l > r) swap(l, r);
//     int k = 0;
//     while ((1 << (k + 1)) <= r - l + 1) ++k;
//     int x = dp[l][k], y = dp[r - (1 << k) + 1][k];
//     return ver[(dep[x] < dep[y]) ? x : y];
// }
```

linear_seq

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int, int> PII;
const ll mod = 1e9 + 7;
```

```
    ll powmod(ll a, ll b) {ll res = 1; a %= mod; assert(b >= 0); for (; b; b >>= 1) {if (b &
1)res = res * a % mod; a = a * a % mod;} return res;}

    namespace linear_seq {
        const int N = 10010;
        ll res[N], base[N], _c[N], _md[N];

        vector<int> Md;
        void mul(ll *a, ll *b, int k) {
            rep(i, 0, k + k) _c[i] = 0;
            rep(i, 0, k) if (a[i]) rep(j, 0, k) _c[i + j] = (_c[i + j] + a[i] * b[j]) % mod;
            for (int i = k + k - 1; i >= k; i--) if (_c[i])
                    rep(j, 0, SZ(Md)) _c[i - k + Md[j]] = (_c[i - k + Md[j]] - _c[i] *
_md[Md[j]]) % mod;
            rep(i, 0, k) a[i] = _c[i];
        }
        int solve(ll n, VI a, VI b) {
            ll ans = 0, pnt = 0;
            int k = SZ(a);
            assert(SZ(a) == SZ(b));
            rep(i, 0, k) _md[k - 1 - i] = -a[i]; _md[k] = 1;
            Md.clear();
            rep(i, 0, k) if (_md[i] != 0) Md.push_back(i);
            rep(i, 0, k) res[i] = base[i] = 0;
            res[0] = 1;
            while ((1ll << pnt) <= n) pnt++;
            for (int p = pnt; p >= 0; p--) {
                mul(res, res, k);
                if ((n >> p) & 1) {
                    for (int i = k - 1; i >= 0; i--) res[i + 1] = res[i]; res[0] = 0;
                    rep(j, 0, SZ(Md)) res[Md[j]] = (res[Md[j]] - res[k] * _md[Md[j]]) % mod;
                }
            }
            rep(i, 0, k) ans = (ans + res[i] * b[i]) % mod;
            if (ans < 0) ans += mod;
            return ans;
        }
        VI BM(VI s) {
            VI C(1, 1), B(1, 1);
            int L = 0, m = 1, b = 1;
            rep(n, 0, SZ(s)) {
                ll d = 0;
                rep(i, 0, L + 1) d = (d + (ll)C[i] * s[n - i]) % mod;
                if (d == 0) ++m;
                else if (2 * L <= n) {
                    VI T = C;
                    ll c = mod - d * powmod(b, mod - 2) % mod;
                    while (SZ(C) < SZ(B) + m) C.pb(0);
                    rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c * B[i]) % mod;
                    L = n + 1 - L; B = T; b = d; m = 1;
                } else {
                    ll c = mod - d * powmod(b, mod - 2) % mod;
                    while (SZ(C) < SZ(B) + m) C.pb(0);
                    rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c * B[i]) % mod;
                    ++m;
                }
            }
```

```
        return C;
    }
    int gao(VI a, ll n) {
        VI c = BM(a);
        c.erase(c.begin());
        rep(i, 0, SZ(c)) c[i] = (mod - c[i]) % mod;
        return solve(n, c, VI(a.begin(), a.begin() + SZ(c)));
    }
};
```

快速乘

```
typedef long long ll;

// mod <= 1e12
inline ll mul(ll a, ll b, ll mod) {
    return (((a * (b >> 20) % mod) << 20) + (a * (b & ((1 << 20) - 1)))) % mod;
}

inline ll mul(ll a, ll b, ll mod) {
    ll d = (ll)floor(a * (long double)b / mod + 0.5);
    ll ret = (a * b - d * mod) % mod;
    if (ret < 0) ret += mod;
    return ret;
}
```

Miller-Rabbin

```
typedef long long ll;

const int psize = 1010000;
bool isp[psize];
int prime[psize], tot;

void prime_table() {
    register int i, j;
    for (i = 2, tot = 0; i < psize; i++) {
        if (!isp[i]) prime[tot++] = i;
        for (j = 0; j < tot && prime[j] * i < psize; j++) {
            isp[prime[j] * i] = true;
            if (i % prime[j] == 0) break;
        }
    }
}

ll qpow(ll a, ll t, ll mod) {
    ll b = 1;
    for (; t > 0; t >>= 1, a = a * a % mod) {
        if (t & 1) {
```

```
            b = b * a % mod;
        }
    }
    return b;
}

bool witness(ll a, ll n) {
    int t = 0;
    ll u = n - 1;
    for (; ~u & 1; u >>= 1) t++;
    ll x = qpow(a, u, n), _x = 0;
    while (t--) {
        _x = mul(x, x, n);
        if (_x == 1 && x != 1 && x != n - 1) return true;
        x = _x;
    }
    return _x != 1;
}

bool Miller(ll n) {
    if (n < 2) return false;
    if (n < psize) return !isp[n];
    if (~n & 1) return false;
    for (int j = 0; j <= 7; j++) {
        if (witness(rand() % (n - 1) + 1, n)) {
            return false;
        }
    }
    return true;
}
```

蒙哥马利取模

```
typedef long long ll;
typedef unsigned long long ull;

using i64 = long long;
using u64 = unsigned long long;
using u128 = __uint128_t;

struct Mod64 {
    Mod64() : n_(0) {}
    Mod64(u64 n) : n_(init(n)) {}
    static u64 modulus() { return mod; }
    static u64 init(u64 w) { return reduce(u128(w) * r2); }
    static void set_mod(u64 m) {
        mod = m;
        assert(mod & 1);
        inv = m;
        for (int i = 0; i < 5; ++i) {
            inv *= 2 - inv * m;
        }
        r2 = -u128(m) % m;
```

```
    }
    static u64 reduce(u128 x) {
        u64 y = u64(x >> 64) - u64((u128(u64(x) * inv) * mod) >> 64);
        return i64(y) < 0 ? y + mod : y;
    }
    Mod64 &operator+=(Mod64 rhs) {
        n_ += rhs.n_ - mod;
        if (i64(n_) < 0) n_ += mod;
        return *this;
    }
    Mod64 operator+(Mod64 rhs) const { return Mod64(*this) += rhs; }
    Mod64 &operator*=(Mod64 rhs) {
        n_ = reduce(u128(n_) * rhs.n_);
        return *this;
    }
    Mod64 operator*(Mod64 rhs) const { return Mod64(*this) *= rhs; }
    u64 get() const { return reduce(n_); }
    static u64 mod, inv, r2;
    u64 n_;
};
u64 Mod64::mod, Mod64::inv, Mod64::r2;
```

Pollard_rho

```
int tot;
long long factor[10000];

long long pollard_rho(long long x, long long c) {
    long long i = 1, k = 2;
    long long x0 = rand() % x, y = x0;
    while (true) {
        i++;
        x0 = (mul(x0, x0, x) + c) % x;
        long long d = __gcd(y - x0, x);
        if (d != 1 && d != x) return d;
        if (y == x0) return x;
        if (i == k) { y = x0, k <<= 1; }
    }
}

void findfac(long long n) {
    if (Miller(n)) {
        factor[tot++] = n;
        return;
    }
    long long p = n;
    while (p >= n) p = pollard_rho(p, rand() % (n - 1) + 1);
    findfac(p), findfac(n / p);
}
```

斯特林公式

$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

莫比乌斯反演

假设对于数论函数f(n)和F(n)，有以下关系式：$F(n) = \sum_{d|n} f(d)$

则将其默比乌斯反转公式定义为：$f(n) = \sum_{d|n} \mu(d) F\left(\frac{n}{d}\right)$