数论

forgottencsc

Sept 21, 2019

数论

初等数论

求 n 以内所有数的所有因子

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

int main() {
    for (int i = 1; i <= n; ++i)
        for (int j = i; j <= n; j += i)
            ds[i].push_back(j);
}
```

整数分解

```
vector<ll> ifd(ll n) {
    vector<ll> v;
    for (ll d = 1; d * d <= n; ++d) {
        if (n % d) continue; v.push_back(d);
        if (d * d != n) v.push_back(n / d);
    }
    return v;
}
```

质因数分解

```
#define W 1000001

bool ip[W]; vector<ll> ps;
void sieve() {
    ps.reserve(W * 1.3 / log(W));
    memset(ip, 1, sizeof(ip)); ip[1] = 0;
    for (int i = 2; i != W; ++i) {
        if (ip[i]) ps.push_back(i);
        for (int p : ps) {
            if (i * p >= W) break;
            ip[i * p] = 0;
            if (i % p == 0) break;
        }
    }
}

vector<pair<ll, ll>> pfd(ll n) {
    vector<pair<ll, ll>> res;
```

```
    for (ll p : ps) {
        if (p * p > n) break;
        if (n % p) continue;
        res.emplace_back(p, 0);
        do res.back().second++;
        while ((n /= p) % p == 0);
    }
    if (n != 1) res.emplace_back(n, 1);
    return res;
}
```

二元一次不定方程

定理：$ax+by=c$ 有解当且仅当 $\gcd(a,b)|c$

先考虑 $c=0$ 的形式，此时存在无穷组形如

$$x=\frac{b}{d}t, y=-\frac{a}{d}t$$

的正整数解。

由裴蜀定理，$ax+by=c$ 有无数组整数解。

扩展欧几里得算法可找出 $ax+by=d$ 的一组特解 (x_0,y_0)，这里 $d=\gcd(a,b)$。

```
// find (u,v) s.t. au+bv=gcd(a,v)
ll exgcd(ll a, ll b, ll& u, ll& v) { ll d;
    if (b) d = exgcd(b, a % b, v, u), v -= (a / b) * u;
    else d = a, u = 1, v = 0; return d;
}
```

于是可得所有解

$$x=\frac{cx_0+bt}{d}, y=\frac{cy_0-at}{d}$$

一次同余方程

定理：$ax\equiv b \mod m$ 有解当且仅当 $\gcd(a,m)\mid b$

解 $ax\equiv b \mod m$ 等价于解二元一次不定方程 $ax+km=b$。

使用扩展欧几里得算法找到一组特解 (x_0,k_0) 后，易得

$$x=\frac{bx_0+mt}{d}, k=\frac{bk_0-at}{d}$$

即

$$x\equiv\frac{bx_0}{d} \mod \frac{m}{d}$$

```
bool lce(ll& a, ll& b, ll& p) {
    ll x, k, d = exgcd(a, p, x, k);
    if (b % d == 0) {
        a = 1; p /= d;
        b = ((x * b / d) % p + p) % p;
    }
    return a == 1;
}
```

一次同余方程组

中国剩余定理：同余方程组

$$\begin{cases} x \equiv b_1 \mod m_1 \\ x \equiv b_2 \mod m_2 \\ \quad\quad \dots \\ x \equiv b_n \mod m_n \end{cases}$$

有解

$$x \equiv \sum_{k=1}^{n} b_k u_k v_k \mod M$$

其中 m_i 两两互质，且

$$M = \prod_{k=1}^{n} m_k, u_k = \frac{M}{m_k}, u_k v_k \equiv 1 \mod m_k$$

扩展中国剩余定理　同余方程组

$$\begin{cases} x \equiv b_1 \mod m_1 \\ x \equiv b_2 \mod m_2 \end{cases}$$

有解当且仅当一次同余方程 $b_1 + km_1 \equiv b_2 \mod m_2$ 有解

```
bool crt(ll& b1, ll& m1, ll b2, ll m2) {
    ll a = m1, b = ((b2 - b1) % m2 + m2) % m2, p = m2;
    if (!lce(a, b, p)) return false;
    else { b1 += b * m1; m1 *= p; return true; }
}
```

二次剩余

勒让德符号：

$$\left(\frac{p}{q}\right) = \begin{cases} 1, p \quad q \\ -1, p \quad q \end{cases}$$

（广义）互倒定律：若 p, q 是两个不同的奇素数，则

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}$$

模质数意义下的二次剩余　a 是模 p 的二次剩余当且仅当

$$a^{\frac{p-1}{2}} \equiv 1 (mod p)$$

模非质数意义下的二次剩余　a 是模 $m = \prod p_i$ 的二次剩余当且仅当 a 是模任意 p_i 的二次剩余。

模意义下的平方根　时间复杂度 $O(\log^2(p))$。

```
// Tonelli-Shanks algorithm. 1s~5e3
ll msqrt(ll n, ll p) {
    if (!n) return 0;
    ll q = p - 1, s = 0, z = 2;
    //while (~q & 1) q >>= 1, s++;
    q >>= (s = __builtin_ctzll(q));
    if (s == 1) return qpm(n, (p + 1) / 4, p);
    while(qpm(z, (p - 1) / 2, p) == 1) ++z;
    ll c = qpm(z, q, p), t = qpm(n, q, p),
       r = qpm(n, (q + 1) / 2, p), m = s;
    while(t % p != 1) {
        ll i = 1; while(qpm(t, 1ll << i, p) != 1) ++i;
        ll b = qpm(c, 1ll << (m - i - 1), p);
        r = r * b % p; c = (b * b) % p;
        t = (t * c) % p; m = i;
    }
    return min(r, p - r); //    r^2=(p-r)^2=n
}
```

解一元二次同余方程

```
//  assert(a && isprime(p) && p > 2);
bool qce(ll a, ll b, ll c, ll p, ll& x1, ll& x2) {
    ll d = ((b * b - 4 * a * c) % p + p) % p;
    if (qpm(d, (p - 1) / 2, p) == p - 1) return false;
    d = msqrt(d, p); a = inv(2 * a % p, p);
    x1 = (p - b + d) * a % p; x2 = (2 * p - b - d) * a % p;
    return true;
}
```

高次不定方程

勾股数　高次不定方程

$$x^2 + y^2 = z^2$$

的所有正整数解被称为勾股数。

通过枚举 $a > b > 0, \gcd(a, b) = 1, 2 \nmid (a + b)$ 可以获得所有本原勾股数，枚举这些勾股数的所有倍数可以获得所有勾股数。

$$\begin{cases} x &= 2ab \\ y &= a^2 - b^2 \\ z &= a^2 + b^2 \end{cases}$$

```
// Primitive Pythagorean Triple
void get() {
    ll sum = 0;
    for (ll b = 1;; ++b) {
        for (ll a = b + 1;; a += 2) {
            if (gcd(a, b) != 1) continue;
            ll x = 2 * a * b;
            ll y = a * a - b * b;
            ll z = a * a + b * b;
            //cout << x << ' ' << y << ' ' << z << endl;
        }
    }
}
```

佩尔方程　佩尔方程是形如

$$x^2 - dy^2 = 1$$

的二次不定方程。

若 $d < -1$ 则只有 $$

```
bool pell_roots(ll d, ll& x, ll& y) {
    static ll a[20000];
    double s = sqrt(d); ll m = s;
    if (m * m == d) return false;
    ll l = 0, b = m, c = 1; a[l++] = m;
    do {
        c = (d - b * b) / c;
        a[l++] = floor((s + b) / c);
        b = a[l - 1] * c - b;
    } while (a[l - 1] != 2 * a[0]);
    ll p = 1, q = 0;
    for (int i = l - 2; i >= 0; --i)
        swap(p, q), p += q * a[i];
    if (l % 2) x = p, y = q;
    else x = 2 * p * p + 1, y = 2 * p * q;
    return true;
}
```

组合数取模

卢卡斯定理：

$$\binom{n}{k} \equiv \binom{\lfloor \frac{n}{p} \rfloor}{\lfloor \frac{k}{p} \rfloor} * \binom{n \mod p}{k \mod p} \mod p$$

```
ll mbinom(ll n, ll k, ll p) {
    static ll f[N];
    for (int i = f[0] = 1; i != p; ++i) f[i] = f[i - 1] * i % p;
    ll ans = 1;
    do {
        if (n % p < k % p) return 0;
        ans = ans * (f[n % p] * inv(f[k % p] * f[(n - k) % p], p) % p) % p;
        n /= p; k /= p;
    } while (n);
    return ans;
}
```

扩展卢卡斯定理

```
ll mfac(ll n, ll p, ll q) {
    if (!n) return 1;
    static map<ll, vector<ll>> m;
    vector<ll>& v = m[p]; if (v.empty()) v.push_back(1);
    for (int i = v.size(); i <= q; ++i)
        v.push_back(v.back() * (i % p ? i : 1) % q);
    return qpm(v[q], n / q, q) * v[n % q] % q * mfac(n / p, p, q) % q;
}

ll mbinom(ll n, ll k, ll p, ll q) {
```

```
    ll c = 0;
    for (ll i = n; i; i /= p) c += i / p;
    for (ll i = k; i; i /= p) c -= i / p;
    for (ll i = n - k; i; i /= p) c -= i / p;
    return mfac(n, p, q) * inv(mfac(k, p, q), q) % q
    * inv(mfac(n - k, p, q), q) % q * qpm(p, c, q) % q;
}

ll mbinom(ll n, ll k, ll m) {
    vector<pair<ll, ll>> ps = pfd(m);
    ll b = 0, w = 1;
    for (pair<ll, ll> pp : ps) {
        ll p = pp.first, q = 1;
        while(pp.second--) q *= p;
        crt(b, w, mbinom(n, k, p, q), q);
    }
    return b;
}
```

原根与离散对数

原根

当模 m 乘法群为循环群时，其生成元被称为原根。

模 m 意义下的原根存在当且仅当 $m = 2, 4, p^t, 2p^t$，其中 p 为奇素数。

当 $p - 1$ 有超过一个大素因子时需要使用 Pollard's Rho 寻找原根。

```
ll primitive_root(ll p) {
    vector<ll> ds; ll n = p - 1;
    for (ll d = 2; d * d <= n; ++d) {
        if (n % d) continue;
        ds.push_back(d);
        while (n % d == 0) n /= d;
    }
    if (n != 1) ds.push_back(n);
    ll g = 1;
    while(1) {
        bool fail = 0;
        for (ll d : ds)
            if (qpm(g, (p - 1) / d, p) == 1)
                fail = 1;
        if (!fail) return g; else g++;
    }
}
```

BSGS

解方程 $a^x \equiv b \mod p$，其中 p 是质数。

复杂度 $O(\sqrt{p})$。

```
// Usage: bsgs.init(a, p); bsgs.solve(b);
struct bsgs_t {
    static const int S = 1 << 19;
    static const int msk = S - 1;
    ll a, p, m, w;
    int c, h[S], g[S], k[S], v[S];
```

```
    int fin(int x) {
        for (int i = h[x & msk]; ~i; i = g[i])
            if (k[i] == x) return v[i];
        return -1;
    }

    void ins(int x, int e) {
        g[c] = h[x & msk]; k[c] = x;
        v[c] = e; h[x & msk] = c++;
    }

    void init(ll a_, ll p_) {
        c = 0; a = a_; p = p_; w = 1;
        m = ceil(sqrt(p));
        memset(h, 0xff, sizeof(h));
        for (int i = 0; i != m; ++i) {
            if (fin(w) == -1) ins(w, i);
            w = w * a % p;
        }
        assert(gcd(w, p) == 1);
        w = inv(w, p);
        //w = qpm(w, p - 2, p);
    }

    int solve(ll b) {
        for (int i = 0; i != m; ++i) {
            int r = fin(b);
            if (r != -1) return i * m + r;
            b = b * w % p;
        }
        return -1;
    }
} bsgs;
```

扩展 BSGS

解方程 $a^x \equiv b \mod p$，a, p 任意，$p = 1$ 可能需要特判。

复杂度 $O(\sqrt{p})$。

```
ll exbsgs(ll a, ll b, ll m) {
    if (b == 1) return 0;
    ll d, w = 1; int c = 0;
    for (ll d; (d = gcd(a, m)) != 1;) {
        if (b % d) return -1;
        b /= d; m /= d; ++c;
        w = (w * (a / d)) % m;
        if (w == b) return c;
    }
    b = b * inv(w, m) % m;
    bsgs.init(a, m);
    ll res = bsgs.solve(b);
    return res == -1 ? -1 : res + c;
}
```

Pohlig-Hellman

解方程 $a^x \equiv b \mod p$，其中 p 是最大质因子较小的的大质数。

设 $p-1=\prod_{i=1}^{n} p_i^{c_i}$，则时间复杂度为 $O(\sum_{i=1}^{n} c_i\sqrt{p_i})$

```
// Solve g^x=a
ll mlog0(ll g, ll a, ll p) {
    vector<pf> pfs = pfd(p - 1);
    ll x = 0, b = 1;
    for (pf f : pfs) {
        ll q = qpm(f.p, f.c, p), w = 1, t = a, r = 0;
        ll h = qpm(g, (p - 1) / f.p, p);
        bsgs.init(h, p, f.p);
        for (int i = 0; i != f.c; ++i) {
            ll z = bsgs.solve(qpm(t, (p - 1) / (w * f.p), p));
            t = mul(t, qpm(qpm(g, w * z, p), p - 2, p), p);
            r += w * z; w *= f.p;
        }
        crt(x, b, r, q);
    }
    return x;
}

// Solve a^x=b(mod p)
ll mlog(ll a, ll b, ll p) {
    ll g = primitive_root(p);
    ll u = mlog0(g, a, p), v = mlog0(g, b, p), m = p - 1;
    if (!lce(u, v, m))
        return -1;
    else
        return v;
}
```

素性测试与大整数分解

Miller-Rabin

```
// Miller-Rabin primality test(Deterministic)
// { 2, 7, 61 } for 2^32
// { 2, 3, 7, 61, 24251 } for 1e16 (except 46856248255981)
// { 2, 325, 9375, 28178, 450775, 9780504, 1795265022 } for 2^64
bool mr(ll n) {
    if (n % 2 == 0) return n == 2;
    if (n < 128) return (0X816D129A64B4CB6E >> (n / 2)) & 1;
    const int l[7] = { 2, 325, 9375, 28178, 450775, 9780504, 1795265022 };
    ll d0 = n - 1; do d0 >>= 1; while(!(d0 & 1));
    for (ll a : l) {
        if (a % n == 0) return true;
        ll d = d0, t = qpm(a, d, n);
        while(d != n - 1 && t != 1 && t != n - 1)
            d <<= 1, t = mul(t, t, n);
        if (t != n - 1 && !(d & 1)) return false;
    }
    return true;
}
```

Pollard's Rho

```
ll pr(ll n) {
    ll x = 0, y = 0, t = 1, q = 1, c = rand() % (n - 1) + 1;
    for (int k = 2;; k <<= 1, y = x, q = 1) {
```

```
        for (int i = 1; i <= k; ++i) {
            x = (mul(x, x, n) + c) % n;
            q = mul(q, abs(x - y), n);
            if (!(i&127) && (t = gcd(q, n) > 1))
                break;
        }
        if (t > 1 || (t = gcd(q, n)) > 1) break;
    }
    return t;
}

void pfd_pr(vector<ll>& ds, ll n) {
    if (mr(n)) return ds.push_back(n);
    ll p = n; while(p >= n) p = pr(n);
    pfd_pr(ds, p); pfd_pr(ds, n / p);
}

struct pf { ll p, c; };
vector<pf> pfd(ll n) {
    vector<ll> v; pfd_pr(v, n);
    vector<pf> res(1, { v[0], 0 });
    sort(v.begin(), v.end());
    for (ll p : v) {
        if (res.back().p != p)
            res.push_back({ p, 1 });
        else res.back().c++;
    }
    return res;
}
```

积性函数与筛法

积性函数

函数 $f(x)$ 是积性函数当且仅当 $\forall a, b, \gcd(a, b) = 1, f(a)f(b) = f(ab)$。

常见的积性函数：

$$e(n) = [n == 1]$$

$$1(n) = 1$$

$$id^k(n) = n$$

```
ll eulerphi(ll n) {
    ll res = 1;
    for (ll p : ps) {
        if (p * p > n) break;
        if (n % p) continue;
        n /= p; res *= (p - 1);
        while (n % p == 0) { n /= p; res *= p; }
    }
    if (n != 1) res *= n - 1;
    return res;
}
```

```
ll moebiusmu(ll n) {
    ll res = 1;
    for (ll p : ps) {
        if (p * p > n) break;
        if (n % p) continue;
        n /= p; res = -res;
        if (n % p == 0) return 0;
    }
    if (n != 1) res = -res;
    return res;
}
```

狄利克雷卷积

两个函数的狄利克雷卷积为

$$(f * g)(n) = \sum_{d|n} f(d)g(\frac{n}{d})$$

常见积性函数的卷积关系：

$

$e = \mu * 1$

$d = 1 * 1$

$\sigma = id * 1$

$\varphi = id * \mu$

欧拉筛

		(1)	(2)	(3)	(4)
最小质因子幂次	pk[i]	0	1	1	pk[i]+1
最小质因子的幂	px[i]	1	i	p	px[i]*p
莫比乌斯 μ 函数	mu[i]	1	-1	-mu[i]	0
欧拉 φ 函数	ph[i]	1	i-1	ph[i]*(j-1)	ph[i]*j
除数函数 $d(i)$	dc[i]	1	2	dc[i]*2	dc[i]+dc[i]/(pk[i]+1)
除数和函数 $\sigma(i)$	ds[i]	1	i+1	ds[i]*(p+1)	ds[i/px[i]]*((px[i]*p*p)-1)/(p-1)
$\sum_{i=1}^{n} i[\gcd(i,n)=1]$	sigma[i]	i * ph[i] / 2			

```
#define W 1000001
bool ip[W]; vector<ll> ps;
void eulersieve() {
    ps.reserve(N * 1.2 / log(N));
    memset(ip, 1, sizeof(ip)); ip[1] = 0;
    // f[1] = (1)
    for (int i = 2; i != N; ++i) {
        if (ip[i]) {
            ps.push_back(i);
            // f[i] = (2)
        }
        for (ll j : ps) {
            if (i * j >= N) break;
            ip[i * j] = 0;
            if (i % j) {
                // f[i * j] = (3)
```

```
            }
            else {
                // f[i * j] = (4)
                break;
            }
        }
    }
}
```

杜教筛

目的：求积性函数 $f(x)$ 的前缀和 $S_f(n)=\sum_{i=1}^{n} f(i)$。

若存在积性函数 g,h 满足 $f*g=h$，且 S_g 与 S_h 能快速求出，则 $S_f(n)$ 能在 $O(n^{\frac{2}{3}})$ 内求出。

推导：

$$S_h(n)=\sum_{i=1}^{n} h(i)=\sum_{i=1}^{n}\sum_{d|i} g(d)f(\frac{i}{d})=\sum_{d=1}^{n} g(d)\sum_{d|i\wedge i\le n} f(\frac{i}{d})=\sum_{d=1}^{n} g(d)\sum_{j=1}^{\lfloor\frac{n}{d}\rfloor} f(j)=\sum_{d=1}^{n} g(d)S_f(\left\lfloor\frac{n}{d}\right\rfloor)$$

右式可数论分块，用线性筛法处理出前 $n^{\frac{2}{3}}$ 个 $S_f(n)$ 可将复杂度降至 $O(n^{\frac{2}{3}})$。

常见的 f 与 g 和 h。

f	g	h	推导
$i^k\varphi(i)$	id^k	id^{k+1}	$$\sum_{d\mid n} d^k\left(\frac{n}{d}\right)^k\varphi(\frac{n}{d})=n^k\sum_{d\mid n}\varphi(\frac{n}{d})=n^{k+1}$$
$i^k\mu(i)$	id^k	e	$$\sum_{d\mid n} d^k\left(\frac{n}{d}\right)^k\mu(\frac{n}{d})=n^k\sum_{d\mid n}\mu(\frac{n}{d})=n^k[n =$$
$\varphi*\mu$	1	φ	$(\varphi*\mu)*1=\varphi*(\mu*1)=\varphi*e=\varphi$

```
namespace mfps_du {

ll m[N], n, s;
function<ll(ll)> sf, sg, sh;

void init(ll n_) {
    n = n_; s = sqrt(n) + 2;
    fill(m, m + s, 0);
}

ll get(ll x) {
    if (x < N) return sf(x);
    ll& sum = m[n / x];
    if (sum) return sum;
    sum = sh(x);
```

```
    for (ll l = 2, r; l <= x; l = r + 1) {
        r = x / (x / l);
        sum = M(sum - M((sg(r) - sg(l - 1)) * get(x / l)));
    }
    return sum;
}

}
```

min25 筛

目的：求积性函数 $f(x)$ 的前缀和 $S_f(n)=\sum_{i=1}^{n} f(i)$。

若 $f(p)$ 可以表达为关于 p 的简单多项式且 $f(p^e)$ 可以快速求，则可以在 $O(\frac{n^{\frac{3}{4}}}{\log n})$ 内求出 $S_f(n)$。

定义：

1. p_i 表示第 i 个质数。
2. m_i 表示 i 的最小质因子。
3. e_i 表示 i 的最小质因子在 i 中的幂次。

共分两步，求出 $g_k(n)=\sum_{p_i\le n} p_i^k$，再用 $g_k(n)$ 求出 $S_f(n)$。

预处理　预处理出 $k>\sqrt{n}$ 时 k 被映射至的位置。

预处理出 $m\le\sqrt{n}$ 时所有的 $s_k(m)=\sum_{i=1}^{m} p_i^k$

第一步：求出 $g_k(n)$　观察到 i^k 是完全积性函数，考虑利用完全积性筛出 $g_k(n)$，即一步步把最小质因子为某个质数的数的 k 次方筛掉。

定义 $g_k(n,j)$ 表示筛掉了最小质因子小于等于 p_j 的所有 i^k 后余下所有 i^k 的和（只筛合数不筛质数）。也可写作

$$g_k(n,j)=\sum_{i=1}^{j} p_i^k+\sum_{i\le n, m_i>p_j} i^k$$

我们要求的 $g_k(n)=g_k(n,\infty)$。经过简单推导可以得到：

$$g_k(n,j)=\begin{cases} g_k(n,j-1) & p_j>\sqrt{n} \\ g_k(n,j-1)-p_j^k\left(g_k(\lfloor\frac{n}{p_j}\rfloor,j-1)-s_k(j)\right) & p_j\le\sqrt{n} \\ \sum_{i=1}^{n} i^k & j=0 \end{cases}$$

注：第 j 步筛掉的是所有最小质因子为 p_j 的合数的 k 次方之和。在第 $j-1$ 步的基础上，所有 n 以内的最小质因子为 p_j 的合数在除以 p_j 后余下部分的 k 次方之和即为 $g(\lfloor\frac{n}{p_j}\rfloor,j-1)$ 减去未被筛去的小于 p_j 的质数的 k 次方之和。

第二步：求出 $S_f(n)$　这一步需要将 $f(n)$ 在第一步中筛去的在合数位置的取值按相反的顺序加回来。

定义 $S(n,j)$ 为将最小质因子小于等于 p_j 的合数位置上的取值筛去后的 $S_f(n)$，即

$$S(n,j)=\sum_{i\le n, m_i>p_j} f(i)$$

枚举所有最小质因子大于等于 p_j 且最小质因子形如 p_k^e 的数

$$S(n,j)=\sum_{k\ge j}\sum_{e} f(p_k^e)\left(1+S(\lfloor\frac{n}{p_k^e}\rfloor,k+1)\right)$$

最终所求即为 $S(n,0)$。

```
namespace mfps_min25 {

bool ip[N]; ll ps[N], pc;
void sieve() {
    fill(ip + 2, ip + N, 1);
    for (int i = 2; i != N; ++i) {
        if (ip[i]) ps[++pc] = i;
        for (int j = 1; j <= pc; ++j) {
            if (i * ps[j] >= N) break;
            ip[i * ps[j]] = 0;
            if (i % ps[j] == 0) break;
        }
    }
}

ll s1(ll x) { x = M(x); return M(M(x * (x + 1)) * inv2); }
ll s2(ll x) { x = M(x); return M(M(x * (x + 1)) * M((2 * x + 1) * inv6)); }

ll n, sq, r, w[N], c, id1[N], id2[N];
ll sp[3][N], g[3][N];

inline ll id(ll x) { return x <= sq ? id1[x] : id2[n / x]; }

void init(ll n_) {
    if (!pc) sieve();
    n = n_; sq = sqrt(n_); c = 0;
    for (r = 1; ps[r] <= sq; ++r);
    for (int i = 1; i <= r; ++i) {
        sp[0][i] = M(sp[0][i - 1] + 1);
        sp[1][i] = M(sp[1][i - 1] + ps[i]);
        sp[2][i] = M(sp[1][i - 1] + M(ps[i] * ps[i]));
    }
    for (ll l = 1, r; l <= n; l = r + 1) {
        ll v = w[++c] = n / l; r = n / v;
        (v <= sq ? id1[v] : id2[n/v]) = c;
        g[0][c] = M(v - 1);
        g[1][c] = M(s1(v) - 1);
        g[2][c] = M(s2(v) - 1);
    }
    for (int i = 1; i <= r; ++i) {
        ll p = ps[i];
        for (int j = 1; j <= c && p * p <= w[j]; ++j) {
            ll k = id(w[j] / p);
            g[0][j] = M(g[0][j] - M(g[0][k] - sp[0][i - 1]));
            g[1][j] = M(g[1][j] - M((p) * M(g[1][k] - sp[1][i - 1])));
            g[2][j] = M(g[2][j] - M(M(p * p) * M(g[2][k] - sp[2][i - 1])));
        }
    }
}

ll get_f(ll p, ll e, ll q) {
    return 114514;
}

ll get_s(ll n, ll i = 0) {
```

```
    if (ps[i] >= n) return 0;
    ll k = id(n), s[3];
    for (int j = 0; j != 3; ++j) s[j] = M(g[j][k] - sp[j][i]);
    ll sum = M(s[2] - s[1] + s[0]);    // f(p) = p^2-p+1
    for (int j = i + 1; j <= r && ps[j] * ps[j] <= n; ++j)
        for (ll q = ps[j], e = 1; q <= n; q *= ps[j], ++e)
            sum = M(sum + M(get_f(ps[j], e, q) * (get_s(n / q, j) + (e != 1))));
    return sum + (i == 0);
}

}
```

顶和底

顶	底
$\left\lceil\frac{a}{b}\right\rceil = \left\lfloor\frac{a-1}{b}\right\rfloor + 1$	$\left\lfloor\frac{a}{b}\right\rfloor = \left\lceil\frac{a+1}{b}\right\rceil - 1$
$a \geq \left\lceil\frac{b}{c}\right\rceil \Leftrightarrow ac \geq b$	$a \leq \left\lfloor\frac{b}{c}\right\rfloor \Leftrightarrow ac \leq b$
$a > \left\lceil\frac{b}{c}\right\rceil \Leftrightarrow ac > b$	$a < \left\lfloor\frac{b}{c}\right\rfloor \Leftrightarrow ac < b$

```
ll sgn(ll x) { return x < 0 ? -1 : x > 0; }
ll ceil(ll b, ll a) { return b / a + (b % a != 0 && sgn(a) * sgn(b) > 0); }
ll floor(ll b, ll a) { return b / a - (b % a != 0 && sgn(a) * sgn(b) < 0); }
```

类欧几里得

主要参考 https://www.cnblogs.com/zzqsblog/p/8904010.html

$$f(k_1, k_2, a, b, c, n) = \sum_{x=0}^{n} x^{k_1} \left\lfloor \frac{ax+b}{c} \right\rfloor^{k_2}$$

若 $a \geq c$ 或 $b \geq c$，令 $a = q_a c + r_a, b = q_b c + r_b$

$$f(k_1, k_2, a, b, c, n) = \sum_{x=0}^{n} x^{k_1} \left(\left\lfloor \frac{r_a x + r_b}{c} \right\rfloor + q_a x + q_b \right)^{k_2}$$

$$= \sum_{i=0}^{k_2} \binom{k_2}{i} \sum_{x=0}^{n} x^{k_1} \left\lfloor \frac{r_a x + r_b}{c} \right\rfloor^{k_2 - i} (q_a x + q_b)^i$$

$$= \sum_{i=0}^{k_2} \binom{k_2}{i} \sum_{j=0}^{i} \binom{i}{j} q_a^j q_b^{i-j} \sum_{x=0}^{n} x^{k_1+j} \left\lfloor \frac{r_a x + r_b}{c} \right\rfloor^{k_2 - i}$$

$$= \sum_{i=0}^{k_2} \binom{k_2}{i} \sum_{j=0}^{i} \binom{i}{j} q_a^j q_b^{i-j} f(k_1 + j, k_2 - i, r_a, r_b, c, n)$$

若 $a = 0$

$$f(k_1, k_2, a, b, c, n) = \sum_{x=0}^{n} x^{k_1} \left\lfloor \frac{b}{c} \right\rfloor^{k_2} = \left\lfloor \frac{b}{c} \right\rfloor^{k_2} \sum_{x=0}^{n} x^{k_1}$$

若 $k_2 = 0$

$$f(k_1, k_2, a, b, c, n) = \sum_{x=0}^{n} x^{k_1}$$

若 $an + b \leq c$

$$f(k_1, k_2, a, b, c, n) = 0$$

否则有 $a \le c$ 且 $b \le c$，进行代换

$$w^{k_2} = \sum_{y=0}^{w-1}((y+1)^{k_2} - y^{k_2})$$

$$f(k_1,k_2,a,b,c,n) = \sum_{x=0}^{n} x^{k_1} \sum_{y=0}^{\lfloor \frac{ax+b}{c} \rfloor - 1} ((y+1)^{k_2} - y^{k_2})$$

杂项

大整数相乘取模

```
typedef unsigned long long ull;
typedef long double ld;
ull mul(ull a, ull b, ull p) {
    ll res = a * b - p * (ull)((ld)a * (ld)b / (ld)M);
    return res + p * (res < 0) - p * (ret >= (ll)p);
}
```

完全数

偶完全数据有形式 $2^{n-1}(2^n-1)$，目前未发现奇完全数。

反素数

```
// 求在因子数最多的前提下最小的数 x
pair<ll, ll> dfs(ll n, int c, int i, ll a, ll b) {
    if (i == 12) return { -a, b };
    else {
        int p = ps[i]; ll q = 1;
        pair<ll, ll> res = { 0, 0 };
        for (int e = 0; e <= c && n / q; ++e, q *= p)
            res = min(res, dfs(n / q, e, i + 1, a * (e + 1), b * q));
        return res;
    }
}

// Usage:
// pair<ll, ll> res = dfs(n, 114514, 0, 1, 1);
// -res.first: d(x)
// res.second: x
```

10^n 以内因子最多的数

n=	x=	d(x)
1	6	4
2	60	12
3	840	32
4	7560	64
5	83160	128
6	720720	240
7	8648640	448
8	73513440	768
9	735134400	1344
10	6983776800	2304
11	97772875200	4032

n=	x=	d(x)
12	963761198400	6720
13	9316358251200	10752
14	97821761637600	17280
15	866421317361600	26880
16	8086598962041600	41472
17	74801040398884800	64512
18	897612484786617600	103680

常见推导与结论

$$d(ij)=\sum_{x|i}\sum_{y|j}[\gcd(x,y)==1]$$

展开 $[gcd(i,n)=1]$

$$\sum_{i=1}^{m} i^k[gcd(i,n)=1]=\sum_{i=1}^{m} i^k\sum_{d|i\wedge d|n}\mu(d)=\sum_{d|n}\mu(d)\sum_{d|i\wedge i\le m} i^k=\sum_{d|n}\mu(d)\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}(dj)^k=\sum_{d|n}\mu(d)d^k\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}j^k$$

当 $m=n$ 时有

$$\sum_{i=1}^{n} i^k[\gcd(i,n)=1]=\sum_{d|n}\mu(d)d^k s_k(\frac{n}{d})$$

2019 南昌邀请赛网络赛 Tsy's number

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\varphi(i)\,\varphi(j^2)\,\varphi(k^3)}{\varphi(i)\,\varphi(j)\,\varphi(k)}\varphi(\gcd(i,j,k))$$

由

$$\frac{\varphi(p^k)}{\varphi(p)}=\frac{\prod p_i^{\mathrm{k}\ _i-1}(p_i-1)}{\prod p_i^{a_i-1}(p_i-1)}=p^{k-1}$$

有原式

$$=\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m}jk^2\varphi(\gcd(i,j,k))=\sum_{d=1}^{m}\varphi(d)\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m}jk^2[\gcd(i,j,k)=d]$$

$$=\sum_{d=1}^{m}\varphi(d)\,d^3\sum_{i=1}^{\lfloor\frac{m}{d}\rfloor}\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}\sum_{k=1}^{\lfloor\frac{m}{d}\rfloor}jk^2[\gcd(i,j,k)=1]=\sum_{d=1}^{m}\varphi(d)\,d^3\sum_{i=1}^{\lfloor\frac{m}{d}\rfloor}\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}\sum_{k=1}^{\lfloor\frac{m}{d}\rfloor}jk^2\sum_{e|i,e|j,e|k}\mu(e)$$

$$=\sum_{d=1}^{m}\varphi(d)\,d^3\sum_{i=1}^{\lfloor\frac{m}{d}\rfloor}\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}\sum_{k=1}^{\lfloor\frac{m}{d}\rfloor}jk^2\sum_{e=1}^{\lfloor\frac{m}{d}\rfloor}\mu(e)\,[e|i]\,[e|j]\,[e|k]$$

$$=\sum_{d=1}^{m}\varphi(d)\,d^3\sum_{e=1}^{\lfloor\frac{m}{d}\rfloor}\mu(e)\sum_{i=1}^{\lfloor\frac{m}{d}\rfloor}[e|i]\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}j\,[e|j]\sum_{k=1}^{\lfloor\frac{m}{d}\rfloor}k^2\,[e|k]$$

$$=\sum_{d=1}^{m}\varphi\left(d\right)d^{3}\sum_{e=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\mu\left(e\right)e^{3}\sum_{i=1}^{\left\lfloor\frac{m}{\mathrm{de}}\right\rfloor}1\sum_{j=1}^{\left\lfloor\frac{m}{\mathrm{de}}\right\rfloor}j\sum_{k=1}^{\left\lfloor\frac{m}{\mathrm{de}}\right\rfloor}k^{2}=\sum_{f=1}^{m}\left(f^{3}\sum_{i=1}^{\left\lfloor\frac{m}{f}\right\rfloor}1\sum_{j=1}^{\left\lfloor\frac{m}{f}\right\rfloor}j\sum_{k=1}^{\left\lfloor\frac{m}{f}\right\rfloor}k^{2}\right)\sum_{d|f}\varphi\left(d\right)\mu\left(\frac{f}{d}\right)$$

$$=\sum_{f=1}^{m}\left[f^{3}s_{0}\left(\left\lfloor\frac{m}{f}\right\rfloor\right)s_{1}\left(\left\lfloor\frac{m}{f}\right\rfloor\right)s_{2}\left(\left\lfloor\frac{m}{f}\right\rfloor\right)\right]\sum_{d|f}\varphi\left(d\right)\mu\left(\frac{f}{d}\right)$$

2019 西安邀请赛 Product　题意：求

$$\prod_{i=1}^{n}\prod_{j=1}^{n}\prod_{k=1}^{n}m^{\gcd(i,j)[k|\gcd(i,j)]}$$

解：

$$\prod_{i=1}^{m}\prod_{j=1}^{m}\prod_{k=1}^{m}w^{\gcd(i,j)[k|\gcd(i,j)]}=\mathrm{pow}\left(w,\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m}\gcd\left(i,j\right)\left[k|\gcd\left(i,j\right)\right]\right)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m}\gcd\left(i,j\right)\left[k|\gcd\left(i,j\right)\right]=\sum_{d=1}^{m}d\sum_{i=1}^{m}\sum_{j=1}^{m}\left[\gcd\left(i,j\right)=d\right]\sum_{k=1}^{m}\left[k|d\right]$$

$$=\sum_{d=1}^{m}d\sigma\left(d\right)\sum_{i=1}^{m}\sum_{j=1}^{m}\left[\gcd\left(i,j\right)=d\right]=\sum_{d=1}^{m}d\sigma\left(d\right)\sum_{i=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\sum_{j=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\left[\gcd\left(i,j\right)=1\right]$$

$$=\sum_{d=1}^{m}d\sigma\left(d\right)\sum_{i=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\sum_{j=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\sum_{e|i,e|j}\mu\left(e\right)=\sum_{d=1}^{m}d\sigma\left(d\right)\sum_{e}^{\left\lfloor\frac{m}{d}\right\rfloor}\mu\left(e\right)\sum_{i=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\sum_{j=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\left[e|i\right]\left[e|j\right]$$

$$=\sum_{d=1}^{m}d\sigma\left(d\right)\sum_{e}^{\left\lfloor\frac{m}{d}\right\rfloor}\mu\left(e\right)\left(\sum_{i=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\left[e|i\right]\right)\left(\sum_{j=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\left[e|j\right]\right)=\sum_{d=1}^{m}d\sigma\left(d\right)\sum_{e=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\mu\left(e\right)\left\lfloor\frac{m}{de}\right\rfloor^{2}$$

$$=\sum_{f=1}^{m}\left\lfloor\frac{m}{f}\right\rfloor^{2}\sum_{d|f}d\sigma\left(d\right)\mu\left(\frac{f}{d}\right)$$

CF1097F Alex and a TV Show　题意：四种操作：

1. 设第 i 个集合为 $\{x\}$
2. 设第 i 个集合为第 j 个集合与第 k 个集合的并
3. 设第 i 个集合为第 j 个集合与第 k 个集合的笛卡儿积并将每个有序对取 gcd 所得集合
4. 求第 i 个集合中 x 的出现次数模 2

定义 $f(S,x)$ 为集合 S 中 x 的出现次数，$g(S,x)$ 为集合 S 中 x 的倍数的出现次数。

$$g(S,x)=\sum_{y\in S}[x|y]=\sum_{x|y}f(S,y)$$

$$f(S,x)=\sum_{y\in S}[x=y]=\sum_{x|y}\mu(\frac{y}{x})g(S,y)$$

操作 2：

$$f(S \cup T, x) = \sum_{y \in S}[x = y] + \sum_{y \in T}[x = y] = f(S, x) + f(T, x)$$

$$g(S \cup T, x) = \sum_{x|y} f(S \cup T, y) = \sum_{x|y}(f(S, y) + f(T, y)) = g(S, x) + g(T, y)$$

操作 3：

$$f(S \times T, x) = \sum_{y_1 \in S}\sum_{y_2 \in T}[x = \gcd(y_1, y_2)]$$

$$g(S \times T, x) = \sum_{y_1 \in S}\sum_{y_2 \in T}[x|y_1][x|y_2] = g(S, x)g(T, x)$$

用 `bitset` 维护 g 即可