PN筛(Power Number Seive)

简介

$$\text{通常用于求积性函数前缀和}: F(n) = \sum_{i=1}^{n} f(i), f\text{为积性函数}$$

做法

$$\text{实质上是用}h\text{和}g\text{取拟合}f$$

$$\text{构造一个函数}g,\text{满足以下条件：}\begin{cases} g\text{是积性函数} \\ g(p) = f(p) \\ G(n) = \sum_{i=1}^{n} g(i)\text{能快速求} \end{cases}$$

$$\text{那么}f = h * g,\text{这里}h\text{也是积性函数}.$$

$$\text{则有：}f(n) = \sum_{d|n} h(d) g(\lfloor \frac{n}{d} \rfloor)$$

$$\begin{aligned} F(n) = \sum_{i=1}^{n} f(i) &= \sum_{i=1}^{n} \sum_{d|i} h(d) g(\lfloor \frac{i}{d} \rfloor) \\ &= \sum_{i=1}^{n} \sum_{d=1}^{n} [d|i] h(d) g(\lfloor \frac{i}{d} \rfloor) \\ &= \sum_{d=1}^{n} \sum_{i=1}^{n} [d|i] h(d) g(\lfloor \frac{i}{d} \rfloor) \\ &= \sum_{d=1}^{n} h(d) \sum_{i=1}^{n} [d|i] g(\lfloor \frac{i}{d} \rfloor) \\ &= \sum_{d=1}^{n} h(d) \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} g(k), i = kd \\ &= \sum_{d=1}^{n} h(d) G(\lfloor \frac{n}{d} \rfloor), --(*) \end{aligned}$$

$$\text{接下来要确定一下}h\text{的性质来求值,根据}g\text{的定义：}$$

$$f(p) = h(1) * g(p) + g(1) * h(p) \to h(p) = 0, h(1) = 1$$

$$\text{由于}h(p) = 0,\text{故}(*)\text{式} \Rightarrow F(n) = \sum_{d=1, d \in PN}^{n} h(d) G(\lfloor \frac{n}{d} \rfloor), --(**)$$

$$\text{这里的}PN\text{是指}d\text{中不包括某质数的一次项,不难知道这样的}d\text{只有}\sqrt{n}\text{个}$$

$$\text{考虑如何求}(**)\text{：考虑转移方程：当前已经算到}x,\text{由于}PN\text{是不含一次项,所以可以枚举}p, c > 1,\text{转移：}x \to xp^c$$

$$h(xp^c) = h(x) h(p^c),\text{那么用筛法即可求解,故我们只需要考虑}h(p^c)$$

$$f(p^c) = \sum_{d|p^c} h(d) g(\lfloor \frac{p^c}{d} \rfloor)$$

$$f(p^c) = h(p^c) g(1) + \sum_{d|p^{c-1}} h(d) g(\lfloor \frac{p^c}{d} \rfloor)$$

$$h(p^c) = \frac{f(p^c) - \sum_{d|p^{c-1}} h(d) g(\lfloor \frac{p^c}{d} \rfloor)}{g(1)} = \frac{f(p^c) - \sum_{e=1}^{c} h(p^{c-e}) g(p^e)}{g(1)}$$

$$\text{根据}g\text{的具体函数性质和各种筛法杜教筛和分块求出}h(p^c)$$

注意

事实上$h(p) = 0$这个条件不一定满足
就跟下面模板里的例题一样,$f(p) = p - 1 + 2 * [p == 2]$,通过$h$反迪利克雷卷积可以知道$h(p) = 2.$
但事实上这里的影响很少,因为根据上面的推导因为$h(p) = 0$所以只有$\sqrt{n}$个地方有值
因此多一个或几个,那数量级应该还是$O(\sqrt{n})$
同时当我们观察到f的特殊点时也不需要对h特殊处理,因为h是通过f和g反卷出来的

相关复杂度证明

$$|\{PN\}| \le \sqrt{n}$$

$$显然\forall x \in PN, x = a^2b^3. 故这样的x有\int_1^{\sqrt{n}} \sqrt[3]{\frac{n}{x^2}}dx = \sqrt{n}$$

预处理

$$预处理筛至少要2 * \sqrt{n} + 1的空间$$

模板

```
//loj 6053
#include<bits/stdc++.h>
#define LL __int128
#define endl '\n'
#define int long long
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
ll gcd(ll x,ll y){return y?gcd(y,x%y):x;}
ll lcm(ll x,ll y){return x/gcd(x,y)*y;}
ll qpow(ll a,ll b,ll p){a%=p; ll ret=1;for(;b;b>>=1,a=a*a%p) if(b&1)
ret=ret*a%p; return ret;}
ll qpow(ll a,ll b){ll ret=1; for(;b;b>>=1,a*=a) if(b&1) ret*=a; return ret;}
ll getInv(ll x,ll p){return qpow(x,p-2,p);}
const int mod = 7 + 1e9;
const int N = 5 + 2e6;
bool np[N];
int tot, pri[N];
int phi[N];
void seive(int n) {
    np[1] = 1; phi[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (!np[i]) {
            pri[++tot] = i;
            phi[i] = i - 1;
        }
        for (int j = 1; j <= tot && i * pri[j] <= n; j++) {
            np[i * pri[j]] = 1;
            if (i % pri[j] == 0) {
                phi[i * pri[j]] = phi[i] * pri[j] % mod;
                break;
            }
            phi[i * pri[j]] = phi[i] * (pri[j] - 1) % mod;
        }
    }
    for (int i = 1; i <= n; i++) {
        phi[i] += phi[i - 1]; phi[i] %= mod;
    }
}
int sh(int x) {
    int y = x + 1;
```

```
    if (x & 1)
        y >>= 1;
    else
        x >>= 1;
    return (x % mod) * (y % mod) % mod;//计算等差数列，注意这里的取模，非常的恶心，我调了两个小时，一定要分开取模之后再乘
}
unordered_map<int, int> s_p;
int n, sqrt_n;
int djs_phi(int x) {
    if (x < sqrt_n) return phi[x];
    if (s_p[x]) return s_p[x];
    int res = sh(x);
    for (int l = 2, r; l <= x; l = r + 1) {
        int d = x / l; r = x / d;
        res -= (r - l + 1) * djs_phi(d) % mod; res %= mod;
    }
    return s_p[x] = res;
}
int h[N][100];
int ans = 0;
void PN(int x, int v, int id) {
    int d = n / x;
    ans += djs_phi(d) * v % mod;
    ans %= mod;
    //if (id > 1 && x > n / pri[id] / pri[id]) return;
    for (int i = id; i <= tot && pri[i] * pri[i] <= d; i++) {
        //if (i > 1 && x > n / pri[i] / pri[i]) break;
        int nx = x * pri[i];
        for (int j = 1; nx <= n; j++, nx *= pri[i]) {
            if (!h[i][j]) { //默认h[i][j]==0表示没取,但事实上某些算过的点的值就为0,可以多开个标记数组来剪枝
                int F = pri[i] ^ j, G = pri[i] - 1;
                for (int k = 1; k <= j; k++, G *= pri[i]) {
                    F = (F - G % mod * h[i][j - k]) % mod;
                }
                h[i][j] = F;
            }
            if (h[i][j]) {
                PN(nx, v * h[i][j] % mod, i + 1);
            }
        }
    }
}

signed main(){
    cin >> n; sqrt_n = 2 * sqrt(n) + 1;//至少要2 * sqrt(n)
    seive(sqrt_n);
    for (int i = 1; i <= tot; i++) h[i][0] = 1;
    ans = 0;
    PN(1, 1, 1);
    cout << (ans + mod) % mod<< endl;
}
```

```
#include<bits/stdc++.h>
#define LL __int128
#define endl '\n'
```

```
#define int long long
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
ll gcd(ll x,ll y){return y?gcd(y,x%y):x;}
ll lcm(ll x,ll y){return x/gcd(x,y)*y;}
ll qpow(ll a,ll b,ll p){a%=p; ll ret=1;for(;b;b>>=1,a=a*a%p) if(b&1) ret=ret*a%p; return ret;}
ll qpow(ll a,ll b){ll ret=1; for(;b;b>>=1,a*=a) if(b&1) ret*=a; return ret;}
ll getInv(ll x,ll p){return qpow(x,p-2,p);}
const int N = 5 + 1e7;
bool isp[N];
int tot, pri[N];
void init(int n) {
    for (int i = 2; i <= n; i++) {
        if (!isp[i]) {
            isp[i] = 1;
            pri[++tot] = i;
        }
        for (int j = 1; j <= tot && i * pri[j] <= n; j++) {
            isp[i * pri[j]] = 1;
            if (i % pri[j] == 0) break;
        }
    }
}
//f(p^c) = c^p
// F(n) = \sum_{d=1,d is PN}^n{h(d)* G(n/d)}
//g(x) = 1, G(x) = x
//h(p) = 0, h(p^c) = f(p^c) - f(p^{c-1}),h(1) = 1
const int mod = 7 + 1e9;
int n;
int ans = 0;
int _h(int p, int c) {
    return (qpow(c, p, mod) - qpow(c - 1, p, mod) + mod) % mod;
}
void PN(int x, int h, int id) {
    int y = n / x;
    ans += y % mod * h % mod;
    ans %= mod;
    for (int i = id; i <= tot && pri[i] * pri[i] <= y; i++) {
        int nx = x * pri[i] * pri[i];
        for (int j = 2; nx <= n; j++, nx *= pri[i]) {
            PN(nx, h * _h(pri[i], j) % mod, i + 1);
            if (nx > n / pri[i]) break; //爆ll
        }
    }
}
signed main(){
    ios :: sync_with_stdio(false), cin.tie(0), cout.tie(0);
    cin >> n;
    int sqrt_n = sqrt(n) + 1;
    init(sqrt_n);
    ans = 0;
    PN(1, 1, 1);
    cout << (ans + mod) % mod << endl;
}
```