

写在最前面：本文部分内容来自网上各大博客或是各类图书，由我个人整理，增加些许见解，仅做学习交流使用，无任何商业用途。因个人实力时间等原因，本文并非完全原创，请大家见谅。

《算法竞赛中的初等数论》（五）正文 0x50筛法（ACM / OI / MO）（十五万字符数论书）

0x50 筛法

0x51 线性筛法

0x51.1 线性筛法求欧拉函数

0x51.2 线性筛求莫比乌斯函数

0x51.3 线性筛求约数个数函数

0x51.4 线性筛求约数和函数

0x52 杜教筛

0x52.1 杜教筛

0x52.2 求欧拉函数前缀和

0x52.3 求莫比乌斯函数前缀和

0x53 Min_25筛

0x54 洲阁筛

《算法竞赛中的初等数论》（五）正文 0x50筛法（ACM / OI / MO）（十五万字符数论书）

《算法竞赛中的初等数论》（信奥 / 数竞 / ACM）前言、后记、目录索引（十五万字符的数论书）

全文目录索引链接：<https://fanfansann.blog.csdn.net/article/details/113765056>

0x50 筛法

0x51 线性筛法

在上面筛质数的时候，我们讲解了如何使用线性筛线性地筛出 $1, \dots, n$ 中的所有质数。

每次只用一个数用小于当前这个数最小质因子的质数去筛其他数，即**保证每个合数 $i \times p_j$ 只都被自己的最小质因子 p_j 筛掉一遍**，所以复杂度保证是 $O(n)$ 的。

我们知道，凡是积性函数，均可由线性筛法来求解。

0x51.1 线性筛法求欧拉函数

我们注意到在线性筛中，每一个合数都是被最小的质因子筛掉，筛法求素数的同时也得到了每个数的最小质因子，这是线性筛求欧拉函数的关键。

1. 考虑 $n = p_j^k$, $\varphi(n)$

显然有 $\varphi(p_j^k) = p_j^k - p_j^{k-1} = p_j^{k-1} \times (p_j - 1)$

2. 考虑 $n = i \times p_j$, $\varphi(n)$, 当 $p_j \mid i$

即 i 含有因子 p_j , 由于 $i = \frac{n}{p_j}$ 即 i 拥有 n 的所有质因子，由唯一分解定理及欧拉函数计算式得：

$$\begin{aligned}\varphi(n) &= n \times \prod_{i=1}^s \frac{p_i - 1}{p_i} \\ &= p_1 \times n' \times \prod_{i=1}^s \frac{p_i - 1}{p_i} \\ &= p_1 \times \varphi(n')\end{aligned}\tag{1}$$

3. 考虑 $n = i \times p_j$, $\varphi(n)$, 当 $p_j \nmid i$

即 i 与 p_j 互质，积性函数显然有性质：

$$\begin{aligned}\varphi(n) &= \varphi(i) \times \varphi(p_j) \\ &= \varphi(i) \times (p_j - 1)\end{aligned}\tag{2}$$

由于我们仅在在线性筛的框架上增加了一些细节，所以时间复杂度依然是 $O(n)$ 的。

Code

```
1 void get_euler(int n)
2 {
3     phi[1] = 1;
4     for(int i = 2; i ≤ n; ++ i) {
5         if(!vis[i]) {
6             primes[ ++ cnt] = i;
7             phi[i] = i - 1;
8         }
9         for(int j = 1; j ≤ cnt && i * primes[j] ≤ n; ++ j) {
10             vis[i * primes[j]] = true;
11             if(i % primes[j] == 0) { // 最小的质因子
12                 phi[i * primes[j]] = phi[i] * primes[j];
13                 break;
14             }
15             phi[i * primes[j]] = phi[i] * (primes[j] - 1);
16         }
17     }
18 }
```

0x51.2 线性筛求莫比乌斯函数

同样考虑三种情况即可。

1. 考虑 $n = p_j^k$, $\mu(n)$

显然有

$$\mu(p_j) = -1$$

$$\mu(p_j^k) = 0, k > 1$$

2. 考虑 $n = i \times p_j$, $\mu(n)$, 当 $p_j \mid i$

$$\mu(n) = 0$$

3. 考虑 $n = i \times p_j$, $\mu(n)$, 当 $p_j \nmid i$

$$\mu(n) = -\mu(i)$$

Code

```
1 void get_mu(int n)
2 {
3     cnt = 0, mu[1] = 1;
4     for(int i = 2; i ≤ n; ++ i) {
5         if(vis[i] == 0){
6             primes[ ++ cnt] = i;
7             mu[i] = -1;
8         }
9         for(int j = 1; j ≤ cnt && i * primes[j] ≤ n; ++ j) {
10             vis[primes[j] * i] = 1;
11             if(i % primes[j] == 0) break;
12             mu[i * primes[j]] -= mu[i];
13             //mi[i * primes[j]] = -mu[i];
14         }
15     }
16     for(int i = 1; i ≤ n; ++ i)
17         sum[i] += sum[i - 1] + mu[i];
18 }
```

0x51.3 线性筛求约数个数函数

约数个数函数： $d(n)$ ：表示 n 的正因子数目。

$$d(n) = \sum_{i|n} 1$$

定理51.3.1：若 $n = \prod_{i=1}^m p_i^{c_i}$, 则 $d_i = \prod_{i=1}^m c_i + 1 = (c_1 + 1) \times \cdots \times (c_m + 1)$

证明：我们知道 $p_i^{c_i}$ 的约数一共有 $p_i^0, p_i^1, \dots, p_i^{c_i}$ 个，即对于每个质因子 p_i 都可以选择其出现 0 次，1 次， \dots ， c_i 次。我们根据乘法原理，可得 n 的约数个数为 $\prod_{i=1}^m c_i + 1$

这里我们规定 d_i 为 i 的约数个数， num_i 表示 i 的最小质因子出现次数。

随机数据下，约数个数的期望是 $O(\ln n)$

Code

```
1 void get_divisor_num(int n) {
2     d[1] = 1;
3     for (int i = 2; i ≤ n; ++i) {
4         if (vis[i] == 0) {
5             vis[i] = 1;
6             primes[ ++ cnt] = i;
7             d[i] = 2, num[i] = 1;
8         }
9         for (int j = 1; j ≤ cnt && i * primes[j] ≤ n; ++ j) {
10             vis[primes[j] * i] = 1;
11             if (i % primes[j] == 0) {
12                 num[i * primes[j]] = num[i] + 1; //这里的primes[j]一定是i的最小质因子
13                 d[i * primes[j]] = d[i] / num[i * primes[j]] * (num[i * primes[j]] + 1);
14                 //数i*primes[j]的质因子primes[j]的出现次数num增加,即公式中的ci增加,对答案的贡献增加,所以需要更新
15                 break;
16             } else num[i * primes[j]] = 1, d[i * primes[j]] = d[i] * 2;
17         }
18     }
19 }
20
```

Ox51.4 线性筛求约数和函数

约数和函数： $\sigma(n)$ ：表示 n 的所有正因子之和。

$$\sigma(n) = \sum_{d|n} d$$

定理51.3.2： 若 $n = \prod_{i=1}^m p_i^{c_i}$ ，则

$$\sigma_i = \prod_{i=1}^m (\sum_{j=0}^{c_i} (p_i)^j) = (1 + p_1 + p_1^2 + \dots + p_1^{c_1}) \times \dots \times (1 + p_m + p_m^2 + \dots + p_m^{c_m})$$

证明略，思路同约数个数函数 $d(n)$ 的证明。

这里我们规定 f_i 表示 i 的约数和， g_i 表示 i 的最小质因子 $p + p^1 + p^2 + \dots + p^k$ 。

```
1 void get_divisor_sum() {
2     g[1] = f[1] = 1;
3     for (int i = 2; i ≤ n; ++ i) {
```



```

4         if (vis[i] == 0) {
5             vis[i] = 1;
6             primes[ ++ cnt] = i;
7             g[i] = i + 1;
8             f[i] = i + 1;
9         }
10        for (int j = 1; j ≤ cnt && i * primes[j] ≤ n; ++ j) {
11            v[i * primes[j]] = 1;
12            if (i % primes[j] == 0) {
13                g[i * primes[j]] = g[i] * primes[j] + 1;
14                f[i * primes[j]] = f[i] / g[i] * g[i * primes[j]]; //更新
15                break;
16            } else {
17                f[i * primes[j]] = f[i] * f[primes[j]];
18                g[i * primes[j]] = primes[j] + 1;
19            }
20        }
21    }
22    for (int i = 1; i ≤ n; ++ i)
23        f[i] = (f[i - 1] + f[i]) % mod;
24 }

```

我们可以利用线性筛求解其他的**所有**积性函数的值，仅需根据具体的性质自行推导两种情况的处理方法即可。

n 的正约数之积.... $n^{\left\lfloor \frac{d(x)}{2} \right\rfloor}$

0x52 杜教筛

0x52.1 杜教筛

杜教筛被用来处理数论函数的前缀和问题。对于一个前缀和，杜教筛可以在低于线性时间的复杂度 ($O(n^{\frac{2}{3}})$) 内求解。

由 $f(n)$ ，计算 $S(n) = \sum_{i=1}^n f(i)$ 要求低于线性，构造出形如 $\lfloor \frac{n}{d} \rfloor$ \longrightarrow 整除分块优化

即：由 $S(n)$ 构造关于 $S(\lfloor \frac{n}{d} \rfloor)$ 的递推式。

构造两个积性函数 h, g ，满足卷积 $h = g * f$

$$\therefore h(i) = \sum_{d|i} g(d) f\left(\frac{i}{d}\right)$$

$$\therefore \sum_{i=1}^n h(i) = \sum_{i=1}^n \sum_{d|i} g(d) f\left(\frac{i}{d}\right)$$

$$= \sum_{d=1}^n g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f\left(\frac{i}{d}\right)$$

$$= \sum_{d=1}^n g(d) \sum_{d|i} f\left(\frac{i}{d}\right)$$

$$= \sum_{d=1}^n g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i)$$

$$= \sum_{d=1}^n g(d) S\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

$$\text{即 } \sum_{i=1}^n h(i) = g(1) \cdot S(n) + \sum_{d=2}^n g(d) S\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

$$g(1) \cdot S(n) = \sum_{i=1}^n h(i) - \sum_{d=2}^n g(d) S\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

此外实际上是对所有 $i \leq n$ 做贡献，交换枚举顺序先枚举 d 提出 g 。

$\because d|i \therefore i$ 为 d 的倍数。
则 d 会对 d 的倍数 $k \cdot d$ 做贡献 (即 i)

贡献为 $g(d) \cdot f\left(\frac{k \cdot d}{d}\right) = g(d) \cdot f(k)$

而在 n 中 d 的倍数为 $d \sim \lfloor \frac{n}{d} \rfloor d$ ，有 $\lfloor \frac{n}{d} \rfloor$ 个。

故得到下式

$$\sum_{i=1}^n \sum_{d|i} g(d) f\left(\frac{i}{d}\right) = \sum_{d=1}^n g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i)$$

$$= \sum_{d=1}^n g(d) S\left(\left\lfloor \frac{n}{d} \right\rfloor\right) \quad (S(n) = \sum_{i=1}^n f(i))$$

此处 i 与 d 无关，可替换

$$g(1) \cdot S(n) = \sum_{i=1}^n h(i) - \sum_{i=2}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right) \longrightarrow \text{杜教筛公式}$$

$$\sum_{i=1}^n \sum_{d|i} g(d) f\left(\frac{i}{d}\right)$$

int sum = 0;

for (i = 1 to n)

for (d | i)

sum += g(d) * f(i/d)

杜教筛变换
等价

$$\sum_{d=1}^n g(d) \sum_{d|i} f\left(\frac{i}{d}\right)$$

int sum = 0;

for (d = 1 to n)

int tmp = 0;

for (i = 1 to n && d | i)

tmp += f(i/d)

sum += g(d) * tmp

https://blog.csdn.net/weixin_45697774

杜教筛其实特别简单，就是一个构造 + 和式转换，利用 *Dirichlet* 卷积构造两个积性函数卷起来，将要求的前缀和 $s(n)$ 构造成 $s(\lfloor \frac{n}{i} \rfloor)$ 的形式，这样我们就可以用整除分块来优化复杂度，可以快速解决一类积性函数的前缀和， n 可以达到 $10^9 \sim 10^{10}$ ，积性函数比如莫比乌斯函数 $\mu(x)$ ，欧拉函数 $\phi(x)$ ，约数和函数 $\sigma_k(x)$ ，约数个数函数 $\sigma(x)$ 等等。

● 基本性质定理

$$\text{性质52.1.1: } g(1)S(n) = \sum_{i=1}^n (f * g)(i) - \sum_{d=2}^n g(d)S\left(\left\lfloor \frac{n}{d} \right\rfloor\right) \quad (\text{其中 } S(n) = \sum_{i=1}^n f(i))$$

• 推导结论:

性质52.2.1: $S_{\mu(x)}(n) = 1 - \sum_{d=2}^n S(\lfloor \frac{n}{d} \rfloor)$ (模板)

性质52.2.2: $S_{\varphi(x)}(n) = \sum_{i=1}^n i - \sum_{d=2}^n S(\lfloor \frac{n}{d} \rfloor)$ (模板)

性质52.2.3: $S_{(n^2\varphi(n))}(n) = \sum_{i=1}^n i^3 - \sum_{d=2}^n d^2 S(\lfloor \frac{n}{d} \rfloor)$ (例题)

Ox52.2 求欧拉函数前缀和

欧拉函数前缀和

$$S(n) = \sum_{i=1}^n \varphi(i) \quad \text{利用欧拉函数性质: } n = \sum_{d|n} \varphi(d)$$

① 直接套用杜教筛公式

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

$$n = \sum_{d|n} \varphi(d) \Rightarrow id = \varphi * I$$

$$\text{则 } h = id, g = I$$

$$\text{杜教筛公式: } g(1) S(n) = \sum_{i=1}^n h(i) - \sum_{i=2}^n g(i) S(\lfloor \frac{n}{i} \rfloor)$$

$$\text{得: } S(n) = \sum_{i=1}^n i - \sum_{i=2}^n S(\lfloor \frac{n}{i} \rfloor)$$

② 自行从头推导

$$= \left(\frac{n(n+1)}{2} - \sum_{i=2}^n S(\lfloor \frac{n}{i} \rfloor) \right) / g(1)$$

由欧拉函数重要性质 $n = \sum_{d|n} \varphi(d)$ 推导出来

$$n = \sum_{d|n} \varphi(d)$$

$$n = \varphi(n) + \sum_{d|n, d < n} \varphi(d)$$

$$\varphi(n) = n - \sum_{d|n, d < n} \varphi(d)$$

$$\sum_{i=1}^n \varphi(i) = \sum_{i=1}^n \left(i - \sum_{d|i, d < i} \varphi(d) \right)$$

$$\sum_{i=1}^n \varphi(i) = \frac{n(n+1)}{2} - \sum_{i=1}^n \sum_{d|i, d < i} \varphi(d)$$

改变枚举变量: 枚举 $\frac{i}{d}$, 即枚举 d 的倍数 i , 且 $i \neq d$. 故从2开始

$$\sum_{i=1}^n \varphi(i) = \frac{n(n+1)}{2} - \sum_{\frac{i}{d}=2}^n \sum_{d=1}^{\lfloor \frac{n}{i} \rfloor} \varphi(d)$$

$$\text{设 } S(n) = \sum_{i=1}^n \varphi(i), \text{ 将 } \frac{i}{d} \text{ 写为 } i$$

$$\text{得 } S(n) = \left(\frac{n(n+1)}{2} - \sum_{i=2}^n S(\lfloor \frac{n}{i} \rfloor) \right) / g(1)$$

https://blog.csdn.net/weixin_45697774

0x52.3 求莫比乌斯函数前缀和

莫比乌斯前缀和

$S(n) = \sum_{i=1}^n \mu(i)$, 利用莫比乌斯函数性质: $\sum_{d|n} \mu(d) = \begin{cases} 1, n=1 \\ 0, n>1 \end{cases}$

① 直接套用杜教筛公式

$$\Rightarrow \sum_{d|n} \mu(d) = [n=1]$$

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

$$\sum_{d|n} \mu(d) = [n=1] \Rightarrow \varepsilon = \mu * I$$

$$\text{故 } h = \varepsilon, g = I$$

$$g(1)S(n) = \sum_{i=1}^n h(i) - \sum_{i=2}^n g(i)S(\lfloor \frac{n}{i} \rfloor)$$

$$\begin{aligned} \text{得: } S(n) &= \sum_{i=1}^n \varepsilon(i) - \sum_{i=2}^n S(\lfloor \frac{n}{i} \rfloor) \\ &= (1 - \sum_{i=2}^n S(\lfloor \frac{n}{i} \rfloor)) / g(1) \end{aligned}$$

② 自行推导(套路各)

$$\sum_{d|n} \mu(d) = [n=1]$$

$$[n=1] = \mu(n) + \sum_{d|n, d < n} \mu(d)$$

$$\mu(n) = [n=1] - \sum_{d|n, d < n} \mu(d)$$

$$\sum_{i=1}^n \mu(i) = 1 - \sum_{i=1}^n \sum_{d|i, d < i} \mu(d)$$

$$\sum_{i=1}^n \mu(i) = 1 - \sum_{\frac{i}{d} \geq 2} \sum_{d=2}^{\lfloor \frac{n}{d} \rfloor} \mu(d)$$

$$\sum_{i=1}^n \mu(i) = \sum_{i=1}^n \mu(i) \text{ 将 } \frac{i}{d} \text{ 写成 } i,$$

$$\text{得 } S(n) = (1 - \sum_{i=2}^n S(\lfloor \frac{n}{i} \rfloor)) / g(1)$$

https://blog.csdn.net/weixin_45697774

Template A. 【模板】杜教筛 (Sum) (P4213)

给定一个正整数, 求

$$ans1 = \sum_{i=1}^n \varphi(i)$$

$$ans2 = \sum_{i=1}^n \mu(i)$$

Code

```
1 //const int N = 5e6 + 10; // n^2/3
2 const int N = 1665703; // n^2/3
3 inline int read()
```

```

4 {
5     register int x=0,f=1;
6     char c=getchar();
7     while(c<'0' || c>'9') {if(c=='-') f=-1;c=getchar();}
8     while(c ≥ '0' && c ≤ '9') x=x*10+c-48,c=getchar();
9     return x*f;
10 }
11
12 int cnt;
13 int primes[N + 7];
14 ll mu[N + 7];
15 bool vis[N + 7];
16 ll phi[N + 7];
17 unordered_map<ll, ll> sum_mu; //数组开不了那么大，所以用哈希表
18 unordered_map<ll, ll> sum_phi;
19
20 inline void init(int n)
21 {
22     vis[0] = vis[1] = 1;
23     mu[1] = phi[1] = 1;
24     for (int i = 2; i ≤ n; ++ i) {
25         if (!vis[i]) {
26             primes[ ++ primes[0]] = i;
27             mu[i] = -1;
28             phi[i] = i - 1;
29         }
30         for (int j = 1; j ≤ primes[0] && i * primes[j] ≤ n; ++ j) {
31             vis[i * primes[j]] = 1;
32             if (i % primes[j] == 0) {
33                 mu[i * primes[j]] = 0;
34                 phi[i * primes[j]] = phi[i] * primes[j];
35                 break;
36             }
37             else {
38                 mu[i * primes[j]] = - mu[i];
39                 phi[i * primes[j]] = phi[i] * phi[primes[j]];
40             }
41         }
42     }
43     for (int i = 1; i ≤ n; ++ i) {
44         mu[i] += mu[i - 1];
45         phi[i] += phi[i - 1];
46     }
47 }
48
49 inline int g_sum(int x) //g的前缀和，这里的g = I(x)//常数函数
50 {

```

```

51     return x;
52 }
53
54 inline int get_sum_mu(int x) // 记忆化搜索
55 {
56     if (x ≤ N) return mu[x]; //预处理
57     //if (sum_mu.find(x) ≠ sum_mu.end()) return sum_mu[x]; //记忆化
58     if(sum_mu[x]) return sum_mu[x];
59     int ans = 1; // 杜教筛中推出的1
60     for (ll l = 2, r; l ≤ x; l = r + 1) { // 整除分块
61         r = x / (x / l);
62         //Σ_i=2 {g(i)*S([n/i])} g不一样, S一样, 然后整除分块
63         ans -= (g_sum(r) - g_sum(l - 1)) * get_sum_mu(x / l);
64     }
65     return sum_mu[x] = ans / g_sum(1); // 最后除以g(1)
66 }
67
68 inline ll get_sum_phi(int x)
69 {
70     if(x ≤ N) return phi[x];
71     //if(sum_phi.find(x) ≠ sum_phi.end()) return sum_phi[x];
72     if(sum_phi[x]) return sum_phi[x];
73
74     ll ans = x * ((ll)x + 1) / 2; //杜教筛中的 n(n + 1) / 2
75     for (ll l = 2, r; l ≤ x; l = r + 1) {
76         r = x / (x / l);
77         ans -= 1ll * (g_sum(r) - g_sum(l - 1)) * get_sum_phi(x / l);
78     }
79     return sum_phi[x] = ans / g_sum(1);
80 }
81
82 int main()
83 {
84     init(N);
85     int t;
86     scanf("%d", &t);
87     while(t -- ) {
88         int n;
89         scanf("%d", &n);
90         printf("%lld %d\n", get_sum_phi(n), get_sum_mu(n));
91     }
92     return 0;
93 }

```

当我们的题目中所给出的积性函数前缀和不太好找到性质使用线性筛 $O(n)$ 求解, 或者数据过大需要更加优秀的时间复杂度时, 我们可以用狄利克雷卷上一个积性函数, 使卷完后的式子的前缀和变得简单, 卷完后的前缀和算出来基本上题目就完成了。

例如：我们遇见 $\varphi(x)$ ，就想办法推式子往 $\sum_{d|n} \varphi(d)$ 上凑。

- 做题时想办法凑出 $\sum_{i=1}^n \sum_{d|i} f(d)$ ，且要求这个式子可以直接求出来。

- 我们转化这个式子 $\sum_{i=1}^n \sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} f(d)$ ，再拆出要求的答案：

$$\sum_{i=1}^n \sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} f(d) = \sum_{i=2}^n \sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} f(d) + \sum_{j=1}^n f(d)$$

- $\sum_{i=2}^n \sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} f(d)$ 这部分往下递归做即可。

- 当 $f(i) = \varphi(i)$ 或者 $\mu(i)$ 时， $\sum_{d|n} f(i)$ 可以直接求出来，但是当 $f(i) = i * u(i)$ 时就不行

了，我们需要卷积一个 $g(i) = i$ ，此时 $\sum_{i=1}^n f * g = \sum_{i=1}^n \sum_{d|i} f(d)g(\frac{i}{d}) = \sum_{i=1}^n \sum_{d|i} u(d) = 1$

。

竞赛例题选讲

Problem A. 简单数学题 (Luogu P3768)

由于出题人懒得写背景了，题目还是简单一点好。

输入一个整数 n 和一个整数 p ，你需要求出：

$$\left(\sum_{i=1}^n \sum_{j=1}^n ij \gcd(i, j) \right) \bmod p$$

其中 $\gcd(a, b)$ 表示 a 与 b 的最大公约数。

https://blog.csdn.net/weixin_45697774

$n \leq 10^{10}$, $5 \times 10^8 \leq p \leq 1.1 \times 10^9$ 且 p 为质数。

Solution

$$\begin{aligned}
& \text{计算: } \left(\sum_{i=1}^n \sum_{j=1}^n i \cdot j \cdot \gcd(i, j) \right) \bmod p \\
& \sum_{i=1}^n \sum_{j=1}^n i \cdot j \cdot \gcd(i, j) \\
& = \sum_{d=1}^n d \sum_{i=1}^n \sum_{j=1}^n i \cdot j \cdot [\gcd(i, j) = d] \\
& = \sum_{d=1}^n d \sum_{i=1}^{\frac{n}{d}} \sum_{j=1}^{\frac{n}{d}} i \cdot j \cdot [\gcd(\frac{i}{d}, \frac{j}{d}) = 1] \quad \left. \begin{array}{l} i = id, j = jd \text{ (} d = \gcd(i, j) \text{ 是 } i, j \text{ 的倍数)} \end{array} \right\} \\
& = \sum_{d=1}^n d \sum_{i=1}^{\frac{n}{d}} \sum_{j=1}^{\frac{n}{d}} i \cdot d \cdot j \cdot d \cdot [\gcd(i, j) = 1] \\
& = \sum_{d=1}^n d^3 \sum_{i=1}^{\frac{n}{d}} \sum_{j=1}^{\frac{n}{d}} i \cdot j \cdot [\gcd(i, j) = 1] \\
& = \sum_{d=1}^n d^3 \sum_{i=1}^{\frac{n}{d}} \sum_{j=1}^{\frac{n}{d}} i \cdot j \sum_{k | \gcd(i, j)} \mu(k) \\
& = \sum_{d=1}^n d^3 \sum_{i=1}^{\frac{n}{d}} \sum_{j=1}^{\frac{n}{d}} i \cdot j \sum_{k=1}^{\frac{n}{d}} \mu(k) [k | i] [k | j] \\
& = \sum_{d=1}^n d^3 \sum_{k=1}^{\frac{n}{d}} \mu(k) \sum_{i=1}^{\frac{n}{d}} \sum_{j=1}^{\frac{n}{d}} i \cdot j [k | i] [k | j] \\
& = \sum_{d=1}^n d^3 \sum_{k=1}^{\frac{n}{d}} \mu(k) \sum_{i=1}^{\frac{n}{d}} \sum_{j=1}^{\frac{n}{d}} i \cdot k \cdot j \cdot k \quad \left. \begin{array}{l} i = ik, j = jk \text{ (} i, j \text{ 是 } k \text{ 的倍数)} \end{array} \right\} \\
& = \sum_{d=1}^n d^3 \sum_{k=1}^{\frac{n}{d}} \mu(k) k^2 \sum_{i=1}^{\frac{n}{d}} \sum_{j=1}^{\frac{n}{d}} i \cdot j \\
& = \sum_{d=1}^n d^3 \sum_{k=1}^{\frac{n}{d}} \mu(k) k^2 \left(\sum_{i=1}^{\frac{n}{d}} i \right)^2
\end{aligned}$$

设 $T = kd$

$$\begin{aligned}
& = \sum_{T=1}^n T^2 \sum_{d | T} \mu\left(\frac{T}{d}\right) d \left(\sum_{i=1}^{\frac{n}{T}} i \right)^2 \\
& = \sum_{T=1}^n T^2 \varphi\left(\frac{T}{d}\right) \left(\sum_{i=1}^{\frac{n}{T}} i \right)^2 \\
& = \sum_{T=1}^n T^2 \varphi(T) \left(\sum_{i=1}^{\frac{n}{T}} i \right)^2 \\
& = \sum_{T=1}^n \varphi(T) T^2 \sum_{i=1}^{\frac{n}{T}} i^2
\end{aligned}$$

$n \leq 10^6$ 考虑杜教筛

$$\text{公式: } g(1) S(n) = \sum_{i=1}^n h(i) - \sum_{i=2}^n g(i) S\left(\frac{n}{i}\right)$$

$$\begin{aligned}
& \text{设 } f(x) = x^2 \varphi(x) \quad S(n) = \sum_{i=1}^n f(i) \\
& h(i) = f * g(i)
\end{aligned}$$

$$= \sum_{d | i} f(d) g\left(\frac{i}{d}\right)$$

$$= \sum_{d | i} d^2 \varphi(d) \times g\left(\frac{i}{d}\right)$$

$$\text{显然 } \sum_{d | i} \varphi(d) = i \quad (\varphi * 1 = id)$$

$$\text{故设 } g(x) = i^2 \quad \text{则: } h(i) = \sum_{d | i} d^2 \varphi(d) \cdot \frac{i^2}{d^2} = \sum_{d | i} \varphi(d) i^2 = i^3$$

故根据杜教筛公式得:

$$S(n) = \frac{\sum_{i=1}^n h(i) - \sum_{i=2}^n g(i) S\left(\frac{n}{i}\right)}{g(1)}$$

$$= \sum_{i=1}^n i^3 - \sum_{i=2}^n i^2 S\left(\frac{n}{i}\right)$$

数论分块 (i^2 的区间前缀和)

$$\text{显然 } \sum_{i=1}^n i^3 = 1^3 + 2^3 + \dots + n^3 = (1+2+3+\dots+n)^2 = \frac{n^2(n+1)^2}{4}$$

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$

https://blog.csdn.net/welxin_45697774

注意一点：这种需要输入模数mod的题，一定要等输入模数之后再初始化 init！！不然会RE...膜0可不就RE了嘛...

Time

$$O(n^{\frac{2}{3}})$$

Code

```

1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;

```

```

4 typedef long long ll;
5
6 const int N = 4641589; // [(10^10)^(2/3)]
7
8 int primes[N + 7], cnt;
9 ll phi[N + 7];
10 bool vis[N + 7];
11 unordered_map<ll, ll> sum_f;
12 int mod, n;
13 int inv4, inv6;
14
15 int qpow(int a, int b)
16 {
17     int res = 1;
18     while(b) {
19         if(b & 1) res = 1ll * res * a % mod;
20         a = 1ll * a * a % mod;
21         b >>= 1;
22     }
23     return res % mod;
24 }
25
26 void init(int n)
27 {
28     phi[1] = 1;
29     for(int i = 2; i ≤ n; ++ i) {
30         if(vis[i] == 0) {
31             primes[ ++ cnt] = i;
32             phi[i] = i - 1;
33         }
34         for(int j = 1; j ≤ cnt && i * primes[j] ≤ n; ++ j) {
35             vis[i * primes[j]] = true;
36             if(i % primes[j] == 0) {
37                 phi[i * primes[j]] = 1ll * phi[i] * primes[j] % mod;
38                 break;
39             }
40             phi[i * primes[j]] = 1ll * phi[i] * phi[primes[j]] % mod;
41         }
42     }
43     for(int i = 1; i ≤ n; ++ i) {
44         phi[i] = (1ll * phi[i - 1] + (1ll * phi[i] * i % mod * i % mod)) % mod;
45     }
46 }
47
48
49
50 inline int g_sum(int n) //i^2

```

```

51 {
52     n %= mod;
53     return 1ll * n * inv6 % mod * (n + 1) % mod * (2 * n + 1) % mod;
54 }
55
56 inline int h_sum(int n) //i^3
57 {
58     n %= mod;
59     return 1ll * n * inv4 % mod * n % mod * (n + 1) % mod * (n + 1) % mod;
60 }
61
62 inline int get_s_sum(int x)
63 {
64     if(x ≤ N - 7) return phi[x];
65     if(sum_f.find(x) ≠ sum_f.end()) return sum_f[x];
66
67     ll ans = h_sum(x);
68     for(ll l = 2, r; l ≤ x; l = r + 1) {
69         r = x / (x / l);
70         ans = (ans - 1ll * (g_sum(r) - g_sum(l - 1) + mod) % mod * get_s_sum(x
/ l) % mod) % mod ;
71     }
72     return sum_f[x] = ans / g_sum(1);
73 }
74
75 void solve(int n)
76 {
77     ll ans = 0;
78     for(int l = 1, r; l ≤ n; l = r + 1) {
79         r = n / (n / l);
80         ans = (ans + (1ll * h_sum(n / l) * (get_s_sum(r) - get_s_sum(l - 1) +
mod) % mod)) % mod;
81     }
82     printf("%lld\n", ans);
83     return ;
84 }
85
86 signed main()
87 {
88
89     scanf("%lld%lld", &mod, &n);
90     init(N - 7);
91     inv4 = qpow(4, mod - 2);
92     inv6 = qpow(6, mod - 2);
93     solve(n);
94     //cout << "ok" << endl;
95     return 0;

```

Problem B. Product (2019 ACM/ICPC 全国邀请赛 (西安) B)

给定 $n(n \leq 10^9), m(m \leq 2 \times 10^9), p(p \leq 2 \times 10^9)$

计算:

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

Solution

计算: $\prod_{i=1}^n \prod_{j=1}^n \prod_{k=1}^n m^{\gcd(i,j)[k|\gcd(i,j)]} \bmod p, n \leq 10^9, m \leq 2 \times 10^9, p \leq 2 \times 10^9, p \text{ 是质数}$

显然直接欧拉降幂, p 是质数: $\gcd(a, p) = 1 \Rightarrow (x^a \bmod p) = x^{a \bmod \varphi(p)} = x^{a \bmod (p-1)}$

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

$$= m^{\left(\sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \gcd(i,j)[k|\gcd(i,j)] \right) \bmod (p-1)} \quad \text{显然连乘} \rightarrow \text{次幂相加}$$

我们仅需计算次幂中的和式, 最后快速幂计算即可.

$$\text{即: } \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \gcd(i,j)[k|\gcd(i,j)] \bmod (p-1)$$

考虑其实际意义为: 所有 $k|\gcd(i,j)$ 即 $\gcd(i,j)$ 为 k 的倍数的个数

显然可以交换枚举顺序, 先枚举 k , 再枚举 k 的倍数 $\gcd(i,j) = x$.

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

$$\sum_{k=1}^n \sum_{k|x} x \sum_{i=1}^n \sum_{j=1}^n [\gcd(i,j)=x] \quad \gcd(i,j)=x, \text{经典套路, 交换位置}$$

$$\sum_{k=1}^n \sum_{k|x} x \sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{x} \rfloor} [\gcd(i,j)=1]$$

分析后面式子的实际含义: $\sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{x} \rfloor} [\gcd(i,j)=1]$ 表示在 $1 \sim \lfloor \frac{n}{x} \rfloor$ 中互质的个数, 因为 $\gcd(i,j)=1$,

所以对于任意两数, 例如 3 互质, 3, 1 互质. 注意枚举到的 i, j , 若 $i=j$ 则只能算一次. 每次 i, j 仅需减一次

$$\text{故 } \sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{x} \rfloor} [\gcd(i,j)=1] = \sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} (2\varphi(i) - 1), \quad n \leq 10^9, \text{直接用杜教筛即可}$$

最后仅需要计算 $\sum_{k=1}^n \sum_{k|x} x$ 即可. 分析其实际意义, 即 k 在 $1 \sim n$ 中的倍数和

显然对于每个 k 和 n 均会存在 $\lfloor \frac{n}{k} \rfloor$ 个 k 的倍数, 显然组成公差为 k 的等差数列. 故

$$\sum_{k=1}^n \sum_{k|x} x = \sum_{x=1}^n x d(x) = \sum_{k=1}^n \frac{k(\lfloor \frac{n}{k} \rfloor + 1) \lfloor \frac{n}{k} \rfloor}{2}$$

$$\text{故最后的答案为 } m^b, \quad b = \sum_{k=1}^n \frac{k(\lfloor \frac{n}{k} \rfloor) \lfloor \frac{n}{k} \rfloor}{2} - \sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} 2\varphi(i) - \sum_{k=1}^n \frac{k(\lfloor \frac{n}{k} \rfloor) \lfloor \frac{n}{k} \rfloor}{2}$$

使用杜教筛可做到 $O(\sqrt{n})$

https://blog.csdn.net/weixin_45897774

Hint

```
1 res = (res + (1ll * (l + r) * (r - l + 1) / 2) % mod * ((1ll + x / l) * (x / l) / 2) % mod) % mod; //AC
2 res = (res + (1ll * (l + r) * (r - l + 1) / 2) % mod * (1ll + x / l) % mod * (x / l) / 2 % mod) % mod; //WA
```

想想看, ^^ (答案就是我脑子抽了)

Time

$$O(\sqrt{n} + n^{\frac{2}{3}})$$

AC Code

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int N = 5e6 + 6;
5 int mod;
6 #define int long long
7 #define mult(x, y) (1ll * x * y ≥ mod ? 1ll * x * y % mod : 1ll * x * y)
8 #define minus(x, y) (1ll * x - y < 0 ? 1ll * x - y + mod : 1ll * x - y)
9 #define plus(x, y) (1ll * x + y ≥ mod ? 1ll * x + y - mod : 1ll * x + y)
10 #define ck(x) (x ≥ mod : x - mod : x)
11 typedef long long ll;
12
13 ll n, m, p;
14
15 ll primes[N], cnt, num[N], d[N];
16 ll phi[N];
17 //ll sum[N]; // x * d(x)
18 bool vis[N];
19
20 unordered_map<ll, ll> M_sum;
21 unordered_map<ll, ll> M_phi;
22
23 void init(ll n)
24 {
25     d[1] = 1;
26     phi[1] = 1;
27     for(ll i = 2; i ≤ n; ++ i) {
28         if(vis[i] == 0) {
29             primes[ ++ cnt] = i;
30             phi[i] = i - 1;
31             d[i] = 2, num[i] = 1;
32         }
33         for(ll j = 1; j ≤ cnt && i * primes[j] ≤ n; ++ j) {
34             vis[i * primes[j]] = true;
35             if(i % primes[j] == 0) {
36                 phi[i * primes[j]] = phi[i] * primes[j];
37                 num[i * primes[j]] = num[i] + 1;
38                 d[i * primes[j]] = (d[i] / num[i * primes[j]] * (num[i *
primes[j]] + 1)) % mod;
39                 break;
40             }
41             phi[i * primes[j]] = phi[i] * (primes[j] - 1);
42             num[i * primes[j]] = 1;

```



```

43         d[i * primes[j]] = (d[i] * 2) % mod;
44     }
45 }
46
47 for(ll i = 1; i ≤ n; ++ i) {
48     phi[i] = (phi[i] + phi[i - 1]) % mod;
49     d[i] = d[i] * i % mod;
50     d[i] = (d[i] + d[i - 1]) % mod;
51 }
52 }
53
54 ll qpow(ll a, ll b, ll mod)
55 {
56     ll res = 1;
57     while(b) {
58         if(b & 1) res = res * a % mod;
59         a = a * a % mod;
60         b >>= 1;
61     }
62     return res;
63 }
64
65 inline int g_sum(int x)
66 {
67     return x;
68 }
69
70 inline ll get_sum_phi(int x)
71 {
72     if(x ≤ N - 7) return phi[x];
73     if(M_phi[x]) return M_phi[x];
74
75     ll ans = mult(x, (1ll * x + 1) / 2);
76     ll res = 0;
77     for(ll l = 2, r; l ≤ x; l = r + 1) {
78         r = x / (x / l);
79         res = plus(res, 1ll * (g_sum(r) - g_sum(l - 1)) * get_sum_phi(x / l))
80         % mod;
81     }
82     return M_phi[x] = minus(ans, res) % mod;
83 }
84
85 inline ll get_sum_sum(ll x)
86 {
87     if(x ≤ N - 7) return d[x];
88     if(M_sum[x]) return M_sum[x];

```

```

89     ll res = 0;
90     for(ll l = 1, r; l ≤ x; l = r + 1) {
91         r = x / (x / l);
92         //sum_k=l^r = 平均值乘上长度 (公差为1的等差数列)
93         res = (res + (1ll * (l + r) * (r - l + 1) / 2) % mod * ((1ll + x / l)
          * (x / l) / 2) % mod) % mod; //AC
94         //res = (res + (1ll * (l + r) * (r - l + 1) / 2) % mod * (1ll + x / l)
          % mod * (x / l) / 2 % mod) % mod; //WA
95     }
96     return M_sum[x] = res;
97 }
98
99 signed main()
100 {
101     scanf("%lld%lld%lld", &n, &m, &p);
102     mod = p - 1;
103     init(N - 7);
104     ll ans = 0;
105     for(ll l = 1, r; l ≤ n; l = r + 1) {
106         r = n / (n / l);
107         ans = plus(ans, 1ll * get_sum_phi(n / l) * minus(get_sum_sum(r),
          get_sum_sum(l - 1)) % mod) % mod;
108     }
109     ans = plus(ans, ans) % mod;
110     ans = minus(ans, get_sum_sum(n));
111     printf("%lld\n", qpow(m, ans, p) % p);
112     return 0;
113 }

```

Problem C. Grisaia (2018ACM四川省赛G)

计算:

$$ans = \sum_{i=1}^n \sum_{j=1}^i (n \bmod (i \times j))$$

其中 $T \leq 5, n \leq 10^{11}$

Solution

使用模的展开式将上述和式展开后, 显然套路枚举 $k = i \times j$, 由于 $n \leq 10^{11}$, 杜教筛即可。

筛出:

$$f(x) = x \times d(x)$$

$$g(x) = x \times \mu(x)$$

然后整除分块即可。

$$\begin{aligned}
 \text{计算: } ans &= \sum_{i=1}^n \sum_{j=1}^i (n \bmod i \times j) \\
 &= \sum_{i=1}^n \sum_{j=1}^i (n - \lfloor \frac{n}{i} \rfloor \times i \times j) \\
 &= n \sum_{i=1}^n \sum_{j=1}^i 1 - \sum_{i=1}^n \sum_{j=1}^i \lfloor \frac{n}{i} \rfloor \times i \times j \\
 &= n \frac{n(n+1)}{2} - \sum_{i=1}^n \sum_{j=1}^i \lfloor \frac{n}{i} \rfloor \times i \times j
 \end{aligned}$$

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

设 $ij=k$, 考虑枚举 k
 $i \leq n, j \leq i$
 若 $i \cdot j > n$, 则 $\lfloor \frac{n}{i} \rfloor = 0$
 故只需枚举 $k \leq n$ 即可

$$\text{即: } \sum_{k=1}^n \lfloor \frac{n}{k} \rfloor \sum_{i=1}^n \sum_{j=1}^i [ij=k] k$$

和式: $\sum_{i=1}^n \sum_{j=1}^i$ 很难处理

考虑统一为: $\sum_{j=1}^n$

显然有 $\sum_{i=1}^n \sum_{j=1}^i [ij=k] \cdot k$

$$= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [ij=k] k + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [ij=k] [i=j] k$$

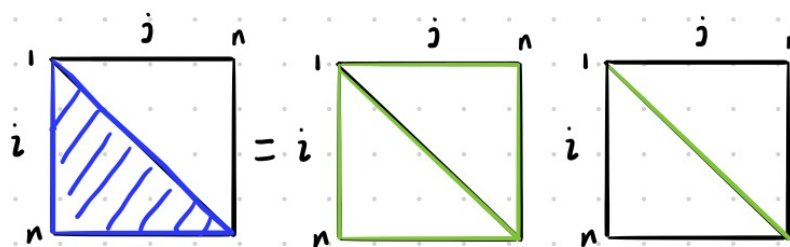
$$= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [ij=k] k + \frac{1}{2} \sum_{i=1}^n [i^2=k] i^2$$

$$\sum_{k=1}^n \lfloor \frac{n}{k} \rfloor \sum_{i=1}^n \sum_{j=1}^i [ij=k] k$$

$$= \sum_{k=1}^n \lfloor \frac{n}{k} \rfloor \cdot \frac{1}{2} \left(\sum_{i=1}^n \sum_{j=1}^n [ij=k] k + \sum_{i=1}^n [i^2=k] i^2 \right)$$

$$\sum_{k=1}^n \lfloor \frac{n}{k} \rfloor \left(\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [ij=k] k \right)$$

$$= \frac{1}{2} \sum_{k=1}^n \lfloor \frac{n}{k} \rfloor \sum_{i=1}^n \sum_{j=1}^n [ij=k] k$$



$$\sum_{i=1}^n \sum_{j=1}^i [ij=k] \cdot k = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [ij=k] k + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [ij=k] [i=j] k$$

(j可等于i)

对角线的值乘了1/2只算了一半

应包含全部对角线 $i=j$ 的值

(可以打表验证)

加上缺的那一半对角线的值

与前面式子一起整除分块即可

$$\text{即: 设 } f(x) = \sum_{k=1}^x \lfloor \frac{x}{k} \rfloor \sum_{i=1}^x [i^2=k] i^2$$

$$\text{整除分块时前缀和: } \text{sum}(x) = \sum_{k=1}^x \sum_{i=1}^x [i^2=k] i^2$$

$$= \sum_{i=1}^{\sqrt{x}} i^2$$

$$= \frac{\sqrt{x}(\sqrt{x}+1)(2\sqrt{x}+1)}{6}$$

$$\sum_{k=1}^n \left\lfloor \frac{n}{k} \right\rfloor \sum_{i=1}^n \sum_{j=1}^n [ij=k] k \longrightarrow \text{显然 } \sum_{i=1}^n \sum_{j=1}^n [ij=k]$$

$$= \sum_{k=1}^n \left\lfloor \frac{n}{k} \right\rfloor k \cdot d(k)$$

的实际含义为 $ij=k$ 的 i, j 的对数 $\rightarrow k$ 的因子数 $\rightarrow \sum_{i|k} 1 = d(k)$

$$\text{设 } f(k) = k d(k)$$

考虑杜教筛, 即构造出 $g(x), h(x)$

$$\text{使得 } h(n) = \sum_{t|n} f(t) g\left(\frac{n}{t}\right)$$

$$\text{根据 } f(t) = t d(t)$$

$$\text{杜教筛公式: } S(n) = \frac{\sum_{i=1}^n h(i) - \sum_{i=2}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)}{g(1)}$$

我们希望构造出的 $h(i)$ 是常数方便计算。

$$\text{显然有 } t \times \frac{n}{t} = n \text{ 为常数}$$

$$\text{故 } g\left(\frac{n}{t}\right) = \frac{n}{t} \quad ()$$

$$\text{显然有 } d * \mu = 1 \Rightarrow \sum_{t|n} d(t) \times \mu\left(\frac{n}{t}\right) = 1$$

$$\text{故 } g(n) = \frac{n}{t} \mu\left(\frac{n}{t}\right)$$

$$\text{则 } h(n) = \sum_{t|n} f(t) g\left(\frac{n}{t}\right)$$

$$= \sum_{t|n} \frac{n}{t} \times \mu\left(\frac{n}{t}\right) \times t \times d(t)$$

$$= \sum_{t|n} n \times \mu\left(\frac{n}{t}\right) \times d(t)$$

$$= n \sum_{t|n} \mu\left(\frac{n}{t}\right) \times d(t)$$

$$= n \times 1$$

$$= n$$

$$S(n) = \frac{\sum_{i=1}^n h(i) - \sum_{i=2}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)}{g(1)}$$

$$= \frac{n(n+1)}{2} - \sum_{i=2}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

$$O(Tn^{\frac{2}{3}})$$

考虑杜教筛计算 $g(i)$ 的前缀和

$$\text{设: } f(x) = x \cdot \mu(x)$$

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

$$\text{同理有 } d \times \frac{n}{d} = n$$

$$\text{故: } g\left(\frac{n}{d}\right) = \frac{n}{d}$$

$$\text{则 } h(n) = \sum_{d|n} f \cdot g$$

$$= \sum_{d|n} d \cdot \mu(d) \cdot \frac{n}{d}$$

$$= \sum_{d|n} \mu(d) n$$

$$= [n=1]$$

$$= 1$$

$$\text{故: } S(n) = \frac{\sum_{i=1}^n h(i) - \sum_{i=2}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)}{g(1)}$$

$$= 1 - \sum_{i=2}^n i \cdot S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

综上所述:

$$\text{ans} = \frac{n(n+1)}{2} \times n - \text{ans2}$$

$$\text{ans2} = \frac{1}{2} \sum_{k=1}^{\sqrt{n}} \left\lfloor \frac{n}{k} \right\rfloor \left(f(k) + \frac{\sqrt{k}(\sqrt{k}+1)(\sqrt{k}+2)}{6} \right)$$

$$f(k) = k d(k)$$

$$g(k) = k \mu(k)$$

$$S(n) = \sum_{i=1}^n f(i)$$

$$= \frac{n(n+1)}{2} - \sum_{i=2}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

$$S_2(n) = \sum_{i=1}^n g(i)$$

$$= 1 - \sum_{i=2}^n i S_2\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

https://blog.csdn.net/weixin_45697774

Hint

注意 $n \leq 10^{11}$, 中间多处会爆 `long long`, 强转成 `__int128` 即可。

Code

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 #define int long long
5 #define ll __int128
6 const int N = 31644346;
```

```

7
8 int n, m;
9 int mu[N];
10 int primes[N], cnt;
11 int d[N];
12 int num[N];
13 unordered_map<int, ll> sum_mui;
14 unordered_map<int, ll> sum_dk;
15 bool vis[N];
16 int sum[N];
17
18 inline ll read()
19 {
20     register ll x = 0, f = 1;
21     char c = getchar();
22     while(c < '0' || c > '9') {if(c == '-') f = -1; c = getchar();}
23     while(c ≥ '0' && c ≤ '9') x = x * 10 + c - 48, c = getchar();
24     return x * f;
25 }
26
27 inline void print(ll x)
28 {
29     if(x < 10)
30     {
31         putchar(x + 48);
32         return;
33     }
34     print(x / 10), print(x % 10);
35 }
36
37 void init(int n)
38 {
39     vis[0] = vis[1] = 1;
40     mu[1] = d[1] = 1;
41     for(int i = 2; i ≤ n; ++ i) {
42         if(vis[i] == 0) {
43             primes[ ++ cnt] = i;
44             mu[i] = -1;
45             d[i] = 2 * i;
46             num[i] = 1;
47         }
48         for(int j = 1; j ≤ cnt && i * primes[j] ≤ n; ++ j) {
49             vis[i * primes[j]] = 1;
50             if(i % primes[j] == 0) {
51                 mu[i * primes[j]] = 0;
52                 num[i * primes[j]] = num[i] + 1;

```

```

53         d[i * primes[j]] = (ll)d[i] / num[i * primes[j]] * (num[i *
primes[j]] + 1) * primes[j];
54         break;
55     }
56     mu[i * primes[j]] -= mu[i];
57     num[i * primes[j]] = 1;
58     d[i * primes[j]] = d[i] * d[primes[j]];
59 }
60 }
61 for(int i = 1; i ≤ n; ++ i) {
62     sum[i] = sum[i - 1] + mu[i] * i;
63     d[i] = d[i - 1] + d[i];
64 }
65 }
66
67 inline ll get_sum_mui(int x)
68 {
69     if(x ≤ N - 7) return sum[x];
70     if(sum_mui.find(x) ≠ sum_mui.end()) return sum_mui[x];
71
72     ll ans = 1;
73     for(ll l = 2, r; l ≤ x; l = r + 1) {
74         r = x / (x / l);
75         ans -= (ll)(r - l + 1) * (l + r) / 2 * get_sum_mui(x / l);
76     }
77     return sum_mui[x] = ans;
78 }
79
80 inline ll get_sum_dk(ll x)
81 {
82     if(x ≤ N - 7) return d[x];
83     if(sum_dk.find(x) ≠ sum_dk.end()) return sum_dk[x];
84     ll ans = x * (x + 1) / 2;
85     for(ll l = 2, r; l ≤ x; l = r + 1) {
86         r = x / (x / l);
87         ans -= (ll)(get_sum_mui(r) - get_sum_mui(l - 1)) * get_sum_dk(x / l);
88     }
89     return sum_dk[x] = ans;
90 }
91
92 ll cal(ll x)
93 {
94     ll limit = sqrt(x + 0.99);
95     ll more = limit * (limit + 1) * (2 * limit + 1) / 6;
96     return (get_sum_dk(x) + more) / 2;
97 }
98

```

```

99 void solve()
100 {
101     ll ans = (ll)n * n * (n + 1) / 2;
102     for(ll l = 1, r; l ≤ n; l = r + 1) {
103         r = n / (n / l);
104         //cout << "ok" << cal(r) - cal(l - 1) * (n / l) << endl;
105         ans -= (ll)(cal(r) - cal(l - 1)) * (n / l);
106     }
107     print(ans);
108     puts("");
109 }
110
111 signed main()
112 {
113     int t;
114     init(N - 7);
115     t = read();
116     while(t -- ) {
117         n = read();
118         solve();
119     }
120     return 0;
121 }

```

ox53 Min_25筛

Min_25筛的博客实在是太难写了，所以就直接摘抄OI-Wiki的讲解吧>_<

author: Marcythm, Xeonacid

link: <https://oiwiki.org/math/number-theory/min-25/>

由于其由 [Min_25](#) 发明并最早开始使用，故称「Min_25 筛」。

从此种筛法的思想方法来说，其又被称为「Extended Eratosthenes Sieve」。

其可以在 $O\left(\frac{n^{\frac{3}{4}}}{\log n}\right)$ 或 $\Theta(n^{1-\epsilon})$ 的时间复杂度下解决一类 **积性函数** 的前缀和问题。

要求： $f(p)$ 是一个关于 p 的项数较少的多项式或可以快速求值； $f(p^c)$ 可以快速求值。

记号

- 如无特别说明，本节中所有记为 p 的变量的取值集合均为全体质数。
- $x/y := \left\lfloor \frac{x}{y} \right\rfloor$
- $\text{isprime}(n) := [\{d : d \mid n\} = 2]$ ，即 n 为质数时其值为 1，否则为 0。
- p_k ：全体质数中第 k 小的质数（如： $p_1 = 2, p_2 = 3$ ）。特别地，令 $p_0 = 1$ 。
- $\text{lpf}(n) := [1 < n] \min\{p : p \mid n\} + [1 = n]$ ，即 n 的最小质因数。特别地， $n = 1$ 时，其值为 1。

- $F_{\text{prime}}(n) := \sum_{2 \leq p \leq n} f(p)$
- $F_k(n) := \sum_{i=2}^n [p_k \leq \text{lpf}(i)] f(i)$

具体方法

观察 $F_k(n)$ 的定义，可以发现答案即为 $F_1(n) + f(1) = F_1(n) + 1$ 。

考虑如何求出 $F_k(n)$ 。通过枚举每个 i 的最小质因子及其次数可以得到递推式：

$$\begin{aligned}
 F_k(n) &= \sum_{i=2}^n [p_k \leq \text{lpf}(i)] f(i) \\
 &= \sum_{k \leq i} \sum_{\substack{c \geq 1 \\ p_i^c \leq n}} f(p_i^c) ([c > 1] + F_{i+1}(n/p_i^c)) + \sum_{\substack{k \leq i \\ p_i \leq n}} f(p_i) \\
 &= \sum_{k \leq i} \sum_{\substack{c \geq 1 \\ p_i^c \leq n}} f(p_i^c) ([c > 1] + F_{i+1}(n/p_i^c)) + F_{\text{prime}}(n) - F_{\text{prime}}(p_{k-1}) \\
 &= \sum_{\substack{k \leq i \\ p_i^2 \leq n}} \sum_{\substack{c \geq 1 \\ p_i^{c+1} \leq n}} (f(p_i^c) F_{i+1}(n/p_i^c) + f(p_i^{c+1})) + F_{\text{prime}}(n) - F_{\text{prime}}(p_{k-1})
 \end{aligned}$$

最后一步推导基于这样一个事实：对于满足 $p_i^c \leq n < p_i^{c+1}$ 的 c ，有 $p_i^{c+1} > n \iff n/p_i^c < p_i < p_{i+1}$ ，故 $F_{i+1}(n/p_i^c) = 0$ 。其边界值即为 $F_k(n) = 0(p_k > n)$ 。

假设现在已经求出了所有的 $F_{\text{prime}}(n)$ ，那么有两种方式可以求出所有的 $F_k(n)$ ：

1. 直接按照递推式计算。
2. 从大到小枚举 p 转移，仅当 $p^2 < n$ 时转移增加值不为零，故按照递推式后缀和优化即可。

现在考虑如何计算 $F_{\text{prime}}(n)$ 。观察求 $F_k(n)$ 的过程，容易发现 F_{prime} 有且仅有 $1, 2, \dots, \lfloor \sqrt{n} \rfloor, n/\sqrt{n}, \dots, n/2, n$ 这 $O(\sqrt{n})$ 处的点值是有用的。一般情况下， $f(p)$ 是一个关于 p 的低次多项式，可以表示为 $f(p) = \sum a_i p^{c_i}$ 。那么对于每个 p^{c_i} ，其对 $F_{\text{prime}}(n)$ 的贡献即为 $a_i \sum_{2 \leq p \leq n} p^{c_i}$ 。分开考虑每个 p^{c_i} 的贡献，问题就转变为了：给定 $n, s, g(p) = p^s$ ，对所有的 $m = n/i$ ，求 $\sum_{p \leq m} g(p)$ 。

Notice: $g(p) = p^s$ 是完全积性函数！

于是设 $G_k(n) := \sum_{i=1}^n [p_k < \text{lpf}(i) \vee \text{isprime}(i)] g(i)$ ，即埃筛第 k 轮筛完后剩下的数的 g 值之和。对于一个合数 x ，必定有 $\text{lpf}(x) \leq \sqrt{x}$ ，则 $F_{\text{prime}} = G_{\lfloor \sqrt{n} \rfloor}$ ，故只需筛到 $G_{\lfloor \sqrt{n} \rfloor}$ 即可。考虑 G 的边界值，显然为 $G_0(n) = \sum_{i=2}^n g(i)$ 。（还记得吗？特别约定了 $p_0 = 1$ ）对于转移，考虑埃筛的过程，分开讨论每部分的贡献，有：

1. 对于 $n < p_k^2$ 的部分， G 值不变，即 $G_k(n) = G_{k-1}(n)$ 。
2. 对于 $p_k^2 \leq n$ 的部分，被筛掉的数必有质因子 p_k ，即 $-g(p_k)G_{k-1}(n/p_k)$ 。
3. 对于第二部分，由于 $p_k^2 \leq n \iff p_k \leq n/p_k$ ，故会有 $\text{lpf}(i) < p_k$ 的 i 被减去。这部分应当加回来，即 $g(p_k)G_{k-1}(p_{k-1})$ 。

则有：

$$G_k(n) = G_{k-1}(n) - [p_k^2 \leq n] g(p_k) (G_{k-1}(n/p_k) - G_{k-1}(p_{k-1}))$$

复杂度分析

对于 $F_k(n)$ 的计算，其第一种方法的时间复杂度被证明为 $O(n^{1-\epsilon})$ （见 zzt 集训队论文 2.3）；对于第二种方法，其本质即为洲阁筛的第二部分，在洲阁论文中也有提及（6.5.4），其时间复杂度被证明为

$$O\left(\frac{n^{\frac{3}{4}}}{\log n}\right).$$

对于 $F_{\text{prime}}(n)$ 的计算，事实上，其实现与洲阁筛第一部分是相同的。考虑对于每个 $m = n/i$ ，只有在枚举满足 $p_k^2 \leq m$ 的 p_k 转移时会对时间复杂度产生贡献，则时间复杂度可估计为：

$$\begin{aligned} T(n) &= \sum_{i^2 \leq n} O\left(\pi\left(\sqrt{i}\right)\right) + \sum_{i^2 \leq n} O\left(\pi\left(\sqrt{\frac{n}{i}}\right)\right) \\ &= \sum_{i^2 \leq n} O\left(\frac{\sqrt{i}}{\ln \sqrt{i}}\right) + \sum_{i^2 \leq n} O\left(\frac{\sqrt{\frac{n}{i}}}{\ln \sqrt{\frac{n}{i}}}\right) \\ &= O\left(\int_1^{\sqrt{n}} \frac{\sqrt{\frac{n}{x}}}{\log \sqrt{\frac{n}{x}}} dx\right) \\ &= O\left(\frac{n^{\frac{3}{4}}}{\log n}\right) \end{aligned}$$

对于空间复杂度，可以发现不论是 F_k 还是 F_{prime} ，其均只在 n/i 处取有效点值，共 $O(\sqrt{n})$ 个，仅记录有效值即可将空间复杂度优化至 $O(\sqrt{n})$ 。

首先，通过一次数论分块可以得到所有的有效值，用一个大小为 $O(\sqrt{n})$ 的数组 lis 记录。对于有效值 v ，记 $\text{id}(v)$ 为 v 在 lis 中的下标，易得：对于所有有效值 v ， $\text{id}(v) \leq \sqrt{n}$ 。

然后分开考虑小于等于 \sqrt{n} 的有效值和大于 \sqrt{n} 的有效值：对于小于等于 \sqrt{n} 的有效值 v ，用一个数组 le 记录其 $\text{id}(v)$ ，即 $\text{le}_v = \text{id}(v)$ ；对于大于 \sqrt{n} 的有效值 v ，用一个数组 ge 记录 $\text{id}(v)$ ，由于 v 过大所以借助 $v' = n/v < \sqrt{n}$ 记录 $\text{id}(v)$ ，即 $\text{ge}_{v'} = \text{id}(v)$ 。

这样，就可以使用两个大小为 $O(\sqrt{n})$ 的数组记录所有有效值的 id 并 $O(1)$ 查询。在计算 $> F_k$ 或 F_{prime} 时，使用有效值的 id 代替有效值作为下标，即可将空间复杂度优化至 $O(\sqrt{n})$ 。

有关代码实现

对于 $F_k(n)$ 的计算，我们实现时一般选择实现难度较低的第一种方法，其在数据规模较小时往往比第二种方法的表现要好；

对于 $F_{\text{prime}}(n)$ 的计算，直接按递推式实现即可。

对于 $p_k^2 \leq n$ ，可以用线性筛预处理出 $s_k := F_{\text{prime}}(p_k)$ 来替代 F_k 递推式中的 $F_{\text{prime}}(p_{k-1})$ 。相应地， G 递推式中的 $G_{k-1}(p_{k-1}) = \sum_{i=1}^{k-1} g(p_i)$ 也可以用此方法预处理。

用 Extended Eratosthenes Sieve 求 **积性函数** f 的前缀和时，应当明确以下几点：

- 如何快速（一般是线性时间复杂度）筛出前 \sqrt{n} 个 f 值；
- $f(p)$ 的多项式表示；
- 如何快速求出 $f(p^c)$ 。

明确上述几点之后按顺序实现以下几部分即可：

1. 筛出 $[1, \sqrt{n}]$ 内的质数与前 \sqrt{n} 个 f 值；
2. 对 $f(p)$ 多项式表示中的每一项筛出对应的 G ，合并得到 F_{prime} 的所有 $O(\sqrt{n})$ 个有用点值；
3. 按照 F_k 的递推式实现递归，求出 $F_1(n)$ 。

例题

求莫比乌斯函数的前缀和

求 $\sum_{i=1}^n \mu(i)$ 。

易知 $f(p) = -1$ 。则 $g(p) = -1, G_0(n) = \sum_{i=2}^n g(i) = -n + 1$ 。直接筛即可得到 F_{prime} 的所有 $O(\sqrt{n})$ 个所需点值。

求欧拉函数的前缀和

求 $\sum_{i=1}^n \varphi(i)$ 。

首先易知 $f(p) = p - 1$ 。对于 $f(p)$ 的一次项 (p) ，有 $g(p) = p, G_0(n) = \sum_{i=2}^n g(i) = \frac{(n+2)(n-1)}{2}$ ；对于 $f(p)$ 的常数项 (-1) ，有 $g(p) = -1, G_0(n) = \sum_{i=2}^n g(i) = -n + 1$ 。筛两次加起来即可得到 F_{prime} 的所有 $O(\sqrt{n})$ 个所需点值。

「LOJ #6053」简单的函数

给定 $f(n)$ ：

$$f(n) = \begin{cases} 1 & n = 1 \\ p \text{ xor } c & n = p^c \\ f(a)f(b) & n = ab \wedge a \perp b \end{cases}$$

易知 $f(p) = p - 1 + 2[p = 2]$ 。则按照筛 φ 的方法筛，对 2 讨论一下即可。此处给出一种 C++ 实现：

```
1 #include <algorithm>
2 #include <cmath>
```



```

3 #include <cstdio>
4
5 const int maxs = 200000; // 2sqrt(n) const int mod = 1000000007;
6
7 template <typename x_t, typename y_t> inline void inc(x_t &x, const
8 y_t &y) { x += y; (mod ≤ x) && (x -= mod); } template <typename
9 x_t, typename y_t> inline void dec(x_t &x, const y_t &y) { x -= y;
10 (x < 0) && (x += mod); } template <typename x_t, typename y_t> inline
11 int sum(const x_t &x, const y_t &y) { return x + y < mod ? x + y :
12 (x + y - mod); } template <typename x_t, typename y_t> inline int
13 sub(const x_t &x, const y_t &y) { return x < y ? x - y + mod : (x -
14 y); } template <typename _Tp> inline int div2(const _Tp &x) { return
15 ((x & 1) ? x + mod : x) >> 1; } //以上目的均为防负数和取模 template <typename
    _Tp>
16 inline long long sqrll(const _Tp &x) { //平方函数 return (long long)x *
17 x; }
18
19 int pri[maxs / 7], lpf[maxs + 1], spri[maxs + 1], pcnt;
20
21 inline void sieve(const int &n) { for (int i = 2; i ≤ n; ++i) {
22     if (lpf[i] == 0) { //记录质数
23         lpf[i] = ++pcnt;
24         pri[lpf[i]] = i;
25         spri[pcnt] = sum(spri[pcnt - 1], i); //前缀和
26     }
27     for (int j = 1, v; j ≤ lpf[i] && (v = i * pri[j]) ≤ n; ++j) lpf[v] = j;
28     } }
29
30 long long global_n; int lim; int le[maxs + 1], // x ≤ \sqrt{n}
31 ge[maxs + 1]; // x > \sqrt{n}
32 #define idx(v) (v ≤ lim ? le[v] : ge[global_n / v])
33
34 int G[maxs + 1][2], Fprime[maxs + 1]; long long lis[maxs + 1]; int
35 cnt;
36
37 inline void init(const long long &n) { for (long long i = 1, j, v; i
38 ≤ n; i = n / j + 1) {
39     j = n / i;
40     v = j % mod;
41     lis[++cnt] = j;
42     (j ≤ lim ? le[j] : ge[global_n / j]) = cnt;
43     G[cnt][0] = sub(v, 1ll);
44     G[cnt][1] = div2((long long)(v + 2ll) * (v - 1ll) % mod); } }
45
46 inline void calcFprime() { for (int k = 1; k ≤ pcnt; ++k) {
47     const int p = pri[k];
48     const long long sqrp = sqrll(p);

```

```

48     for (int i = 1; lis[i] ≥ sqrp; ++i) {
49         const long long v = lis[i] / p;
50         const int id = idx(v);
51         dec(G[i][0], sub(G[id][0], k - 1));
52         dec(G[i][1], (long long)p * sub(G[id][1], spri[k - 1]) % mod);
53     }    /* F_prime = G_1 - G_0 */    for (int i = 1; i ≤ cnt; ++i)
        Fprime[i] = sub(G[i][1], G[i][0]); }
54
55 inline int f_p(const int &p, const int &c) {    /* f(p^{c}) = p xor c
56 */    return p xor c; }
57
58 int F(const int &k, const long long &n) {    if (n < pri[k] || n ≤ 1)
59 return 0;    const int id = idx(n);    long long ans = Fprime[id] -
60 (spri[k - 1] - (k - 1));    if (k == 1) ans += 2;    for (int i = k; i
61 ≤ pcnt && sqrll(pri[i]) ≤ n; ++i) {
62     long long pw = pri[i], pw2 = sqrll(pw);
63     for (int c = 1; pw2 ≤ n; ++c, pw = pw2, pw2 *= pri[i])
64         ans +=
65             ((long long)f_p(pri[i], c) * F(i + 1, n / pw) + f_p(pri[i], c + 1))
66             %
67             mod;    }    return ans % mod; }
68
69 int main() {    scanf("%lld", &global_n);    lim = sqrt(global_n);    //上限
70    sieve(lim + 1000);    //预处理    init(global_n);    calcFprime();
71    printf("%lld\n", (F(1, global_n) + 1ll + mod) % mod);
72
73    return 0; } ``

```

0x54 洲阁筛

author: Early0v0

link: <https://oiwiki.org/math/number-theory/zhou/>

定义

洲阁筛是一种能在亚线性时间复杂度内求出大多数积性函数前缀和的筛法。

下面将以求解 $\sum_{i=1}^n f(i)$ 为例，具体阐述洲阁筛的原理。

约定

- \mathbb{P} 表示质数集, p_i 表示第 i 个质数。
- m 表示 \sqrt{n} 内的质数个数。

##

要求

当 $p \in \mathbb{P}, c \in \mathbb{N}$ 时, $f(p^c)$ 为一个关于 p 的低阶多项式。

思想

- 对于任意 $[1, n]$ 内的整数, 其至多只有一个 $> \sqrt{n}$ 的质因子。
- 利用 $\left\lfloor \frac{n}{i} \right\rfloor (i \in [1, n] \cap \mathbb{N})$ 只有 \sqrt{n} 级别个取值的性质来降低时间复杂度。

思路

将 $[1, n]$ 内的所有整数按是否有 $> \sqrt{n}$ 的质因子分为两类:

$$\sum_{i=1}^n f(i) = \sum_{i=1}^n [\exists d \in (\sqrt{n}, n] \cap \mathbb{P}, d \mid i] f(i) + \sum_{i=1}^n [\forall d \in (\sqrt{n}, n] \cap \mathbb{P}, d \nmid i] f(i)$$

对于前半部分, 枚举最大因子, 根据积性函数的性质可以转换:

$$\sum_{i=1}^n f(i) = \sum_{i=1}^{\sqrt{n}} f(i) \cdot \left(\sum_{d=\lfloor \sqrt{n} \rfloor + 1}^{\lfloor \frac{n}{i} \rfloor} [d \in \mathbb{P}] f(d) \right) + \sum_{i=1}^n [\forall d \in (\sqrt{n}, n] \cap \mathbb{P}, d \nmid i] f(i)$$

前后部分可以分别计算。

Part 1

$$\text{计算 } \sum_{i=1}^{\sqrt{n}} f(i) \cdot \left(\sum_{d=\lfloor \sqrt{n} \rfloor + 1}^{\lfloor \frac{n}{i} \rfloor} [d \in \mathbb{P}] f(d) \right).$$

考虑枚举 i , 然后 $\mathcal{O}(1)$ 计算括号内部分。

记 $g(t, l) = \sum_{i=1}^l [\forall j \in [1, t], \gcd(i, p_j) = 1] f(i)$, 即 $[1, l]$ 中与 p_1, p_2, \dots, p_t 均互质的数的 f 值之和。

这样 Part 1 的计算就变成了 $\sum_{i=1}^{\sqrt{n}} f(i) \cdot g\left(m, \left\lfloor \frac{n}{i} \right\rfloor\right)$ 。

边界 $g(0, l) = \sum_{i=1}^l f(i)$, 转移 $g(t, l) = g(t-1, l) - f(p_t) \cdot g\left(t-1, \left\lfloor \frac{l}{p_t} \right\rfloor\right)$ 。

l 共有 \sqrt{n} 级别种取值, 对于每种取值则需要枚举其质因子, 所以复杂度为 $\mathcal{O}\left(\frac{\sqrt{n}}{\ln \sqrt{n}} \cdot \sqrt{n}\right) = \mathcal{O}\left(\frac{n}{\log n}\right)$, 需要优化。

注意到 $p_{t+1}^2 > l$ 时符合条件的数只有 1, 所以此时 $g(t, l) = f(1) = 1$ 。

代入递推式可得: 当 $p_t^2 > l$ 时, $g(t, l) = g(t-1, l) - f(p_t)$ 。

所以一旦发现 $p_t^2 > l$ 就停止转移, 记此时的 t 为 t_l , 则

$\forall t > t_l, g(t, l) = g(t_l, l) - \sum_{i=t_l}^{t-1} f(p_i)$ 。

预处理质数的 f 值前缀和即可快速求出 g , 时间复杂度被优化至 $\mathcal{O}\left(\frac{n^{\frac{3}{4}}}{\log n}\right)$ 。

Part 2

计算 $\sum_{i=1}^n [\forall d \in (\sqrt{n}, n] \cap \mathbb{P}, d \nmid i] f(i)$ 。

记 $h(t, l) = \sum_{i=1}^l \left[i = \prod_{j=t}^m p_j^{c_j}, c_j \in \mathbb{N} \right] f(i)$, 即 $[1, l]$ 中所有只含 p_t, p_{t+1}, \dots, p_m 质因子的数的 f 值之和。

Part 2 即为求 $h(0, n)$ 。

边界 $h(m+1, l) = 1$, 转移 $h(t, l) = h(t+1, l) + \sum_{c \in \mathbb{N}^*} f(p_t^c) \cdot h\left(t+1, \left\lfloor \frac{l}{p_t^c} \right\rfloor\right)$ 。

l 共有 \sqrt{n} 级别种取值, 所以直接转移复杂度为 $\mathcal{O}\left(\sqrt{n} \cdot \frac{\sqrt{n}}{\ln \sqrt{n}}\right) = \mathcal{O}\left(\frac{n}{\log n}\right)$, 需要优化。

与 g 的优化方式类似, 注意到 $p_t > l$ 时, 能用 p_t, p_{t+1}, \dots, p_m 组成的数只有 1, 此时的 $h(t, l) = f(1) = 1$ 。

类似的, 推出 $\forall p_t^2 > l, h(t, l) = h(t-1, l) + f(p_t)$ 。

所以一旦发现 $p_t^2 > l$ 就停止转移, 记此时的 t 为 t_l , 之后用到 h 时, 把此时的 h 值加上 $\sum_{i=p_{t_l}}^{\min(l, \sqrt{n})} [i \in \mathbb{P}] f(i)$ 即可。

时间复杂度被优化至 $\mathcal{O}\left(\frac{n^{\frac{3}{4}}}{\log n}\right)$ 。

求和

###

算出了 Part 1 和 Part 2 的答案, 将其相加即为 $\sum_{i=1}^n f(i)$ 。

参考

[积性函数线性筛/杜教筛/洲阁筛学习笔记 | Bill Yang's Blog](#)