杜教筛

积性函数

在数论题目中，常常需要根据一些 **积性函数** 的性质，求出一些式子的值。

积性函数：对于函数 $f(n)$ 有 $f(1)=1$ 且对于所有互质的 a 和 b 总有 $f(ab)=f(a)f(b)$ 则称 $f(n)$ 为积性函数。

常见的积性函数有：

- $\sigma_0(n)=d(n)=\sum_{d\mid n }1$
- $\sigma(n)=\sum_{d\mid n}d$
- $\varphi(n)=\sum_{i=1}^n[\gcd(x,i)=1]$
- $\mu(n)=\begin{cases}1&\ n=1\\(-1)^k&\ \prod_{i=1}^k q_i=1\\0 &\ \max\{q_i\}>1 \end{cases}$

积性函数有如下性质：

若 $f(n)$ $g(n)$ 为积性函数，则：

- $h(n)=f(n^p)$
- $h(n)=f^p(n)$
- $h(n)=f(n)g(n)$
- $h(n)=\sum_{d\mid n}f(d)g(\frac n d)$

中的 $h(n)$ 也为积性函数。

在莫比乌斯反演的题目中，往往要求出一些数论函数的前缀和，利用 **杜教筛** 可以快速求出这些前缀和。

杜教筛

杜教筛被用来处理数论函数的前缀和问题。对于求解一个前缀和，杜教筛可以在低于线性时间的复杂度内求解

对于数论函数 f 要求我们计算 $S(n)=\sum_{i=1}^nf(i)$.

我们想办法构造一个 $S(n)$ 关于 $S\left(\left\lfloor\frac n i\right\rfloor\right)$ 的递推式

对于任意一个数论函数 g 必满足 $$ \sum_{i=1}^{n}\sum_{d \mid i}f(d)g\left(\frac{i}{d}\right)=\sum_{i=1}^{n}g(i)S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\ \Leftrightarrow \sum_{i=1}^{n}(f\ast g)(i)=\sum_{i=1}^{n}g(i)S\left(\left\lfloor\frac{n}{i}\right\rfloor\right) $$

略证：

$f(d)g(\frac i d)$ 就是对所有 $i\le n$ 的做贡献，因此变换枚举顺序，枚举 $d,\frac i d$ 分别对应新的 i,j $$ \begin{split} &\sum_{i=1}^n\sum_{d\mid i}f(d)g\left(\frac i d\right)\\ =&\sum_{i=1}^n\sum_{j=1}^{\left\lfloor\frac n i\right\rfloor}g(i)f(j)\\ =&\sum_{i=1}^ng(i)\sum_{j=1}^{\left\lfloor\frac n i\right\rfloor}f(j)\\

=&\sum_{i=1}^ng(i)S\left(\left\lfloor\frac n i\right\rfloor\right) \end{split} $$ 那么可以得到递推式 $$ g(1)S(n)=\sum_{i=1}^n(f*g)(i)-\sum_{i=2}^ng(i)S\left(\left\lfloor\frac n i\right\rfloor\right) $$ 那么假如我们可以快速对 $\sum_{i=1}^n(f*g)(i)$ 求和，并用数论分块求解 $\sum_{i=2}^ng(i)S\left(\left\lfloor\frac n i\right\rfloor\right)$就可以在较短时间内求得 $g(1)S(n)$

问题一

> P4213【模板】杜教筛(Sum)
>
> 题目大意：求 $S_1(n)=\sum_{i=1}^n\mu(i)$ 和 $S_2(n)=\sum_{i=1}^n\varphi(i)$ 的值$n\le 2^{31}-1$

莫比乌斯函数前缀和

由 **狄利克雷卷积**，我们知道：

$\because\varepsilon=\mu*1(\varepsilon(n)=[n=1])$

$\therefore\varepsilon(n)=\sum_{d\mid n}\mu(d)$

$S_1(n)=\sum_{i=1}^n\varepsilon(i)-\sum_{i=2}^nS_1\left(\left\lfloor\frac n i\right\rfloor\right)$

$=1-\sum_{i=2}^nS_1\left(\left\lfloor\frac n i\right\rfloor\right)$

观察到 $\lfloor\frac n i\rfloor$ 最多只有 $O(\sqrt n)$ 种取值，我们就可以应用 **整除分块**（或称数论分块）来计算每一项的值了。

直接计算的时间复杂度为 $O(n^{\frac 3 4})$考虑先线性筛预处理出前 $n^{\frac 2 3}$ 项，剩余部分的时间复杂度为 $$ O(\int_0^{n^{\frac 1 3}}\sqrt{\frac n x}~dx)=O(n^{\frac 2 3}) $$ 对于较大的值，需要用 `map` 存下其对应的值，方便以后使用时直接使用之前计算的结果。

欧拉函数前缀和

当然也可以用杜教筛求出 $\varphi(x)$ 的前缀和，但是更好的方法是应用莫比乌斯反演：

$\sum_{i=1}^n\sum_{j=1}^n1[\gcd(i,j)=1]=\sum_{i=1}^n\sum_{j=1}^n\sum_{d\mid i,d\mid j}\mu(d)\\ =\sum_{d=1}^n\mu(d)\lfloor\frac n d\rfloor^2$

观察到，只需求出莫比乌斯函数的前缀和，就可以快速计算出欧拉函数的前缀和了。时间复杂度 $O(n^{\frac 2 3})$

使用杜教筛求解

求 $S(i)=\sum_{i=1}^n\varphi(i)$

同样地$\varphi*1=Id$ $$ \begin{split} &\sum_{i=1}^n(\varphi\ast 1)(i)=\sum_{i=1}^n1\cdot S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\ &\sum_{i=1}^nID(i)=\sum_{i=1}^n1\cdot S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\

&\frac{1}{2}n(n+1)=\sum_{i=1}^nS\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\
&S(n)=\frac{1}{2}n(n+1)-\sum_{i=2}^nS\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\ \end{split}
$$ 参考代码：

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <map>
using namespace std;
const int maxn = 2000010;
typedef long long ll;
ll T, n, pri[maxn], cur, mu[maxn], sum_mu[maxn];
bool vis[maxn];
map<ll, ll> mp_mu;
ll S_mu(ll x) {
  if (x < maxn) return sum_mu[x];
  if (mp_mu[x]) return mp_mu[x];
  ll ret = 1ll;
  for (ll i = 2, j; i <= x; i = j + 1) {
    j = x / (x / i);
    ret -= S_mu(x / i) * (j - i + 1);
  }
  return mp_mu[x] = ret;
}
ll S_phi(ll x) {
  ll ret = 0ll;
  for (ll i = 1, j; i <= x; i = j + 1) {
    j = x / (x / i);
    ret += (S_mu(j) - S_mu(i - 1)) * (x / i) * (x / i);
  }
  return ((ret - 1) >> 1) + 1;
}
int main() {
  scanf("%lld", &T);
  mu[1] = 1;
  for (int i = 2; i < maxn; i++) {
    if (!vis[i]) {
      pri[++cur] = i;
      mu[i] = -1;
    }
    for (int j = 1; j <= cur && i * pri[j] < maxn; j++) {
      vis[i * pri[j]] = true;
      if (i % pri[j])
        mu[i * pri[j]] = -mu[i];
      else {
        mu[i * pri[j]] = 0;
        break;
      }
    }
  }
```

```
  for (int i = 1; i < maxn; i++) sum_mu[i] = sum_mu[i - 1] + mu[i];
  while (T--) {
    scanf("%lld", &n);
    printf("%lld %lld\n", S_phi(n), S_mu(n));
  }
  return 0;
}
```

问题二

> LuoguP3768简单的数学题
>
> 题目大意：求 $
> \sum_{i=1}^n\sum_{j=1}^ni\cdot j\cdot\gcd(i,j)\pmod{p}
> $ 其中 $n\le 10^{10},5\times 10^8\le p\le 1.1\times 10^9$，$p$ 是质数。

利用 $\varphi * 1=Id$ 做莫比乌斯反演化为 $$ \sum_{d=1}^nF^2\left(\left\lfloor\frac n d\right\rfloor\right)\cdot d^2\varphi(d)\left(F(n)=\frac 1 2n(n+1)\right) $$ 对 $\sum_{d=1}^nF\left(\left\lfloor\frac n d\right\rfloor\right)^2$ 做数论分块，$d^2\varphi(d)$ 的前缀和用杜教筛处理。 $$ f(n)=n^2\varphi(n)=(Id^2\varphi)(n)\\
S(n)=\sum_{i=1}^nf(i)=\sum_{i=1}^n(Id^2\varphi)(i) $$ 需要构造积性函数 g，使得 $f*g$ 和 g 能快速求和。

单纯的 φ 的前缀和可以用 $\varphi*1$ 的杜教筛处理，但是这里的 f 多了一个 Id^2，那么我们就卷一个 Id^2 上去，让它变成常数。 $$ S(n)=\sum_{i=1}^n\left((ID^2\varphi)\ast ID^2\right)(i)-\sum_{i=2}^nID^2(i)S\left(\left\lfloor\frac{n}{i}\right\rfloor\right) $$ 化一下卷积。 $$ \begin{split} &(ID^2\varphi)\ast ID^2)(i)\\ =&\sum_{d \mid i}(ID^2\varphi)(d)ID^2\left(\frac{i}{d}\right)\\ =&\sum_{d \mid i}d^2\varphi(d)\left(\frac{i}{d}\right)^2\\ =&\sum_{d \mid i}i^2\varphi(d)=i^2\sum_{d \mid i}\varphi(d)\\ =&i^2(\varphi\ast1)(i)=i^3 \end{split} $$ 再化一下 $S(n)$。 $$ \begin{split} S(n)&=\sum_{i=1}^n\left((ID^2\varphi)\ast ID^2\right)(i)-\sum_{i=2}^nID^2(i)S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\ &=\sum_{i=1}^ni^3-\sum_{i=2}^ni^2S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\ &=\left(\frac{1}{2}n(n+1)\right)^2-\sum_{i=2}^ni^2S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\ \end{split} $$ 分块求解即可：

参考代码：

```
#include <cmath>
#include <cstdio>
#include <map>
using namespace std;
const int N = 5e6, NP = 5e6, SZ = N;
long long n, P, inv2, inv6, s[N];
int phi[N], p[NP], cnt, pn;
bool bp[N];
map<long long, long long> s_map;
long long ksm(long long a, long long m) {  // 求逆元用
  long long res = 1;
  while (m) {
    if (m & 1) res = res * a % P;
```

```
    a = a * a % P, m >>= 1;
  }
  return res;
}
void prime_work(int k) {  //线性筛 phi□s
  bp[0] = bp[1] = 1, phi[1] = 1;
  for (int i = 2; i <= k; i++) {
    if (!bp[i]) p[++cnt] = i, phi[i] = i - 1;
    for (int j = 1; j <= cnt && i * p[j] <= k; j++) {
      bp[i * p[j]] = 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]];
    }
  }
  for (int i = 1; i <= k; i++)
    s[i] = (1ll * i * i % P * phi[i] % P + s[i - 1]) % P;
}
long long s3(long long k) {
  return k %= P, (k * (k + 1) / 2) % P * ((k * (k + 1) / 2) % P) % P;
}  //立方和
long long s2(long long k) {
  return k %= P, k * (k + 1) % P * (k * 2 + 1) % P * inv6 % P;
}  //平方和
long long calc(long long k) {  //计算 S(k)
  if (k <= pn) return s[k];
  if (s_map[k]) return s_map[k];  //对于超过 pn 的用 map 离散存储
  long long res = s3(k), pre = 1, cur;
  for (long long i = 2, j; i <= k; i = j + 1)
    j = k / (k / i), cur = s2(j),
    res = (res - calc(k / i) * (cur - pre) % P) % P, pre = cur;
  return s_map[k] = (res + P) % P;
}
long long solve() {
  long long res = 0, pre = 0, cur;
  for (long long i = 1, j; i <= n; i = j + 1)
    j = n / (n / i), cur = calc(j),
    res = (res + (s3(n / i) * (cur - pre)) % P) % P, pre = cur;
  return (res + P) % P;
}
int main() {
  scanf("%lld%lld", &P, &n);
  inv2 = ksm(2, P - 2), inv6 = ksm(6, P - 2);
  pn = (long long)pow(n, 0.666667);  // n^(2/3)
  prime_work(pn);
  printf("%lld", solve());
  return 0;
}  //不要为了省什么内存把数组开小..... 卡了好几次80
```