莫比乌斯反演

引入

莫比乌斯反演是数论中的重要内容。对于一些函数 $f(n)$，如果很难直接求出它的值，而容易求出其倍数和或约数和 $g(n)$，那么可以通过莫比乌斯反演简化运算，求得 $f(n)$ 的值。

开始学习莫比乌斯反演前，需要先学习一些前置知识：数论分块、狄利克雷卷积。

莫比乌斯函数

定义

μ 为莫比乌斯函数，定义为

$$\mu(n)=\begin{cases}1 & n=1\\ 0 & n\text{ 含有平方因子}\\ (-1)^k & k\text{ 为 }n\text{ 的本质不同质因子个数}\end{cases}$$

详细解释一下：

令 $n=\prod_{i=1}^k p_i^{c_i}$，其中 p_i 为质因子，$c_i\ge 1$。上述定义表示：

1. $n=1$ 时，$\mu(n)=1$；
2. 对于 $n\ne 1$ 时：
    a. 当存在 $i\in[1,k]$，使得 $c_i>1$ 时，$\mu(n)=0$，也就是说只要某个质因子出现的次数超过一次，$\mu(n)$ 就等于 0；
    b. 当任意 $i\in[1,k]$，都有 $c_i=1$ 时，$\mu(n)=(-1)^k$，也就是说每个质因子都仅仅只出现过一次时，即 $n=\prod_{i=1}^k p_i$，$\{p_i\}_{i=1}^k$ 中个元素唯一时，$\mu(n)$ 等于 -1 的 k 次幂，此处 k 指的便是仅仅只出现过一次的质因子的总个数。

性质

莫比乌斯函数不仅是积性函数，还有如下性质：

$$\sum_{d\mid n}\mu(d)=\begin{cases}1 & n=1\\ 0 & n\ne 1\end{cases}$$

即 $\sum_{d\mid n}\mu(d)=\varepsilon(n)$，$\mu*1=\varepsilon$

证明

设 $n=\prod_{i=1}^{k}{p_i}^{c_i}, n'=\prod_{i=1}^{k}p_i$

那么 $\sum_{d\mid n}\mu(d)=\sum_{d\mid n'}\mu(d)=\sum_{i=0}^{k}\binom{k}{i}\cdot(-1)^i=(1+(-1))^k$

根据二项式定理，易知该式子的值在 $k=0$ 即 $n=1$ 时值为 1 否则为 0，这也同时证明了 $\sum_{d\mid n}\mu(d)=[n=1]=\varepsilon(n)$ 以及 $\mu*1=\varepsilon$

注

这个性质意味着，莫比乌斯函数在狄利克雷生成函数中，等价于黎曼函数 ζ 的倒数。所以在狄利克雷卷积中，莫比乌斯函数是常数函数 1 的逆元。

补充结论

反演结论：$[\gcd(i,j)=1]=\sum_{d\mid\gcd(i,j)}\mu(d)$

直接推导：如果看懂了上一个结论，这个结论稍加思考便可以推出：如果 $\gcd(i,j)=1$ 的话，那么代表着我们按上个结论中枚举的那个 n 是 1，也就是式子的值是 1，反之，有一个与 $[\gcd(i,j)=1]$ 相同的值：0

利用 ε 函数：根据上一结论，$[\gcd(i,j)=1]=\varepsilon(\gcd(i,j))$，将 ε 展开即可。

线性筛

由于 μ 函数为积性函数，因此可以线性筛莫比乌斯函数（线性筛基本可以求所有的积性函数，尽管方法不尽相同）。

线性筛实现

C++

```
void getMu() {
  mu[1] = 1;
  for (int i = 2; i <= n; ++i) {
    if (!flg[i]) p[++tot] = i, mu[i] = -1;
    for (int j = 1; j <= tot && i * p[j] <= n; ++j) {
      flg[i * p[j]] = 1;
      if (i % p[j] == 0) {
        mu[i * p[j]] = 0;
        break;
      }
      mu[i * p[j]] = -mu[i];
    }
  }
}
```

Python

```
def getMu():
mu[1] = 1
for i in range(2, n + 1):
    if flg[i] != 0:
        p[tot] = i; tot = tot + 1; mu[i] = -1
    j = 1
    while j <= tot and i * p[j] <= n:
        flg[i * p[j]] = 1
        if i % p[j] == 0:
            mu[i * p[j]] = 0
            break
        mu[i * p[j]] = mu[i * p[j]] - mu[i]
        j = j + 1
```

拓展

证明

$$\varphi * 1 = \mathrm{id}$$

将 n 分解质因数：$n = \prod_{i=1}^{k} {p_i}^{c_i}$

首先，因为 φ 是积性函数，故只要证明当 $n' = p^c$ 时 $\varphi * 1 = \sum_{d|n'} \varphi(\frac{n'}{d}) = \mathrm{id}$ 成立即可。

因为 p 是质数，于是 $d = p^0, p^1, p^2, \cdots, p^c$

易知如下过程：

$$\begin{aligned}\varphi * 1 &= \sum_{d|n} \varphi(\frac{n}{d}) \\ &= \sum_{i=0}^{c} \varphi(p^i) \\ &= 1 + p^0 \cdot (p-1) + p^1 \cdot (p-1) + \cdots + p^{c-1} \cdot (p-1) \\ &= p^c \\ &= \mathrm{id}\end{aligned}$$

该式子两侧同时卷 μ 可得 $\varphi(n) = \sum_{d|n} d \cdot \mu(\frac{n}{d})$

莫比乌斯变换

设 $f(n), g(n)$ 为两个数论函数。

形式一：如果有 $f(n) = \sum_{d|n} g(d)$，那么有 $g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$。

这种形式下，数论函数 $f(n)$ 称为数论函数 $g(n)$ 的莫比乌斯变换，数论函数 $g(n)$ 称为数论函数 $f(n)$ 的莫比乌斯逆变换（反演）。

容易看出，数论函数 $g(n)$ 的莫比乌斯变换，就是将数论函数 $g(n)$ 与常数函数 1 进行狄利克雷卷积。

注

根据狄利克雷卷积与狄利克雷生成函数的对应关系，数论函数 $g(n)$ 的莫比乌斯变换对应的狄利克雷生成函数，就是数论函数 $g(n)$ 的狄利克雷生成函数与黎曼函数 ζ 的乘积。

形式二：如果有 $f(n) = \sum_{n|d} g(d)$，那么有 $g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$。

证明

方法一：对原式做数论变换。

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

用 $\sum_{d|n} g(d)$ 来替换 $f(\frac{n}{d})$，再变换求和顺序。最后一步变换的依据：$\sum_{d|n} \mu(d) = [n=1]$，因此在 $\frac{n}{k} = 1$ 时第二个和式的值才为 1。此时 $n = k$，故原式等价于 $\sum_{k|n} [n=k] \cdot g(k) = g(n)$

方法二：运用卷积。

原问题为：已知 $f=g*1$，证明 $g=f*\mu$

易知如下转化：$f*\mu=g*1*\mu \implies f*\mu=g$（其中 $1*\mu=\varepsilon$）。

对于第二种形式：

类似上面的方法一，我们考虑逆推这个式子。

$$\begin{aligned}&\sum_{n|d}\mu(\frac{d}{n})f(d)\\=&\sum_{k=1}^{+\infty}\mu(k)f(kn)=\sum_{k=1}^{+\infty}\mu(k)\sum_{kn|d}g(d)\\=&\sum_{n|d}g(d)\sum_{k|\frac{d}{n}}\mu(k)=\sum_{n|d}g(d)\epsilon(\frac{d}{n})\\=&g(n)\end{aligned}$$

我们把 d 表示为 kn 的形式，然后把 f 的原定义代入式子。

发现枚举 k 再枚举 kn 的倍数可以转换为直接枚举 n 的倍数再求出 k，发现后面那一块其实就是 ϵ, 整个式子只有在 $d=n$ 的时候才能取到值。

问题形式

「HAOI 2011」Problem b

求值（多组数据）

$$\sum_{i=x}^{n}\sum_{j=y}^{m}[\gcd(i,j)=k]\qquad(1\leqslant T,x,y,n,m,k\leqslant 5\times 10^4)$$

根据容斥原理，原式可以分成 4 块来处理，每一块的式子都为

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

考虑化简该式子

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

因为 $\gcd(i,j)=1$ 时对答案才用贡献，于是我们可以将其替换为 $\varepsilon(\gcd(i,j))$（$\varepsilon(n)$ 当且仅当 $n=1$ 时值为 1 否则为 0），故原式化为

$$\sum_{i=1}^{\lfloor\frac{n}{k}\rfloor}\sum_{j=1}^{\lfloor\frac{m}{k}\rfloor}\varepsilon(\gcd(i,j))$$

将 ε 函数展开得到

$$\sum_{i=1}^{\lfloor\frac{n}{k}\rfloor}\sum_{j=1}^{\lfloor\frac{m}{k}\rfloor}\sum_{d\mid\gcd(i,j)}\mu(d)$$

变换求和顺序，先枚举 $d\mid\gcd(i,j)$ 可得

$$\sum_{d=1}\mu(d)\sum_{i=1}^{\lfloor\frac{n}{k}\rfloor}[d\mid i]\sum_{j=1}^{\lfloor\frac{m}{k}\rfloor}[d\mid j]$$

易知 $1\sim\lfloor\dfrac{n}{k}\rfloor$ 中 d 的倍数有 $\lfloor\dfrac{n}{kd}\rfloor$ 个，故原式化为

$$\sum_{d=1}^{\min(\lfloor\frac{n}{k}\rfloor,\lfloor\frac{m}{k}\rfloor)}\mu(d)\lfloor\frac{n}{kd}\rfloor\lfloor\frac{m}{kd}\rfloor$$

很显然，式子可以数论分块求解。

时间复杂度 $\Theta(N+T\sqrt{n})$

代码实现

```cpp
#include <algorithm>
#include <iostream>
using namespace std;
constexpr int N = 50000;
int mu[N + 5], p[N + 5];
bool flg[N + 5];

void init() {
  int tot = 0;
  mu[1] = 1;
  for (int i = 2; i <= N; ++i) {
    if (!flg[i]) {
      p[++tot] = i;
      mu[i] = -1;
    }
    for (int j = 1; j <= tot && i * p[j] <= N; ++j) {
      flg[i * p[j]] = true;
      if (i % p[j] == 0) {
        mu[i * p[j]] = 0;
        break;
      }
      mu[i * p[j]] = -mu[i];
    }
  }
  for (int i = 1; i <= N; ++i) mu[i] += mu[i - 1];
}

int solve(int n, int m) {
  int res = 0;
  for (int i = 1, j; i <= min(n, m); i = j + 1) {
    j = min(n / (n / i), m / (m / i));
    res += (mu[j] - mu[i - 1]) * (n / i) * (m / i);  // 代推出来的
式子
  }
  return res;
}

int main() {
  cin.tie(nullptr)->sync_with_stdio(false);
  int T, a, b, c, d, k;
  init();  // 预处理mu数组
  cin >> T;
  for (int i = 1; i <= T; i++) {
    cin >> a >> b >> c >> d >> k;
    // 根据容斥原理，1<=x<=b&&1<=y<=d范围中的答案数减去
1<=x<=b&&1<=y<=c-1范围中的答案数和
    //   1<=x<=a-1&&1<=y<=d范围中的答案数再加上1<=x<=a-1&&1<=y<=c-1
范围中的答案数
    //   即可得到a<=x<=b&&c<=y<=d范围中的答案数
```

```cpp
    // 这一步如果不懂可以画坐标图进行理解
    cout << solve(b / k, d / k) - solve(b / k, (c - 1) / k) -
                solve((a - 1) / k, d / k) + solve((a - 1) / k, (c - 1) / k)
         << '\n';
  }
  return 0;
}
```

「SPOJ 5971」LCMSUM

求值（多组数据）

$$\sum_{i=1}^{n} \operatorname{lcm}(i,n) \quad \text{s.t. } 1 \leqslant T \leqslant 3\times 10^5, 1 \leqslant n \leqslant 10^6$$

易得原式即

$$\sum_{i=1}^{n} \frac{i\cdot n}{\gcd(i,n)}$$

将原式复制一份并且颠倒顺序，然后将 n 一项单独提出，可得

$$\frac{1}{2}\cdot\left(\sum_{i=1}^{n-1}\frac{i\cdot n}{\gcd(i,n)}+\sum_{i=n-1}^{1}\frac{i\cdot n}{\gcd(i,n)}\right)+n$$

根据 $\gcd(i,n)=\gcd(n-i,n)$，可将原式化为

$$\frac{1}{2}\cdot\left(\sum_{i=1}^{n-1}\frac{i\cdot n}{\gcd(i,n)}+\sum_{i=n-1}^{1}\frac{i\cdot n}{\gcd(n-i,n)}\right)+n$$

两个求和式中分母相同的项可以合并。

$$\frac{1}{2}\cdot\sum_{i=1}^{n-1}\frac{n^2}{\gcd(i,n)}+n$$

即

$$\frac{1}{2}\cdot\sum_{i=1}^{n}\frac{n^2}{\gcd(i,n)}+\frac{n}{2}$$

可以将相同的 $\gcd(i,n)$ 合并在一起计算，故只需要统计 $\gcd(i,n)=d$ 的个数。当 $\gcd(i,n)=d$ 时，$\gcd(\frac{i}{d},\frac{n}{d})=1$，所以 $\gcd(i,n)=d$ 的个数有 $\varphi(\frac{n}{d})$ 个。

故答案为

$$\frac{1}{2}\cdot\sum_{d\mid n}\frac{n^2\cdot\varphi(\frac{n}{d})}{d}+\frac{n}{2}$$

变换求和顺序，设 $d'=\frac{n}{d}$，合并公因式，式子化为

$$\frac{1}{2}n\cdot\left(\sum_{d'|n}d'\cdot\varphi(d')+1\right)$$

设 $\mathrm{g}(n)=\sum_{d|n}d\cdot\varphi(d)$，已知 g 为积性函数，于是可以 $\Theta(n)$ 筛出。每次询问 $\Theta(1)$ 计算即可。

下面给出这个函数筛法的推导过程：

首先考虑 $\mathrm{g}(p_j^k)$ 的值，显然它的约数只有 $p_j^0,p_j^1,\cdots,p_j^k$，因此

$$\mathrm{g}(p_j^k)=\sum_{w=0}^{k}p_j^w\cdot\varphi(p_j^w)$$

又有 $\varphi(p_j^w)=p_j^{w-1}\cdot(p_j-1)$，则原式可化为

$$\sum_{w=0}^{k}p_j^{2w-1}\cdot(p_j-1)$$

于是有

$$\mathrm{g}(p_j^{k+1})=\mathrm{g}(p_j^k)+p_j^{2k+1}\cdot(p_j-1)$$

那么，对于线性筛中的 $\mathrm{g}(i\cdot p_j)(p_j|i)$，令 $i=a\cdot p_j^w(\gcd(a,p_j)=1)$，可得

$$\mathrm{g}(i\cdot p_j)=\mathrm{g}(a)\cdot\mathrm{g}(p_j^{w+1})$$
$$\mathrm{g}(i)=\mathrm{g}(a)\cdot\mathrm{g}(p_j^w)$$

即

$$\mathrm{g}(i\cdot p_j)-\mathrm{g}(i)=\mathrm{g}(a)\cdot p_j^{2w+1}\cdot(p_j-1)$$

同理有

$$\mathrm{g}(i)-\mathrm{g}(\frac{i}{p_j})=\mathrm{g}(a)\cdot p_j^{2w-1}\cdot(p_j-1)$$

因此

$$\mathrm{g}(i\cdot p_j)=\mathrm{g}(i)+\left(\mathrm{g}(i)-\mathrm{g}(\frac{i}{p_j})\right)\cdot p_j^2$$

时间复杂度：$\Theta(n+T)$

代码实现

```cpp
#include <iostream>
constexpr int N = 1000000;
int tot, p[N + 5];
long long g[N + 5];
bool flg[N + 5];  // 标记数组

void solve() {
  g[1] = 1;
  for (int i = 2; i <= N; ++i) {
    if (!flg[i]) {
      p[++tot] = i;
      g[i] = (long long)1 * i * (i - 1) + 1;
    }
    for (int j = 1; j <= tot && i * p[j] <= N; ++j) {
      flg[i * p[j]] = true;
      if (i % p[j] == 0) {
        g[i * p[j]] =
            g[i] + (g[i] - g[i / p[j]]) * p[j] * p[j];  // 代入推出来的式子
        break;
      }
      g[i * p[j]] = g[i] * g[p[j]];
    }
  }
}

using std::cin;
using std::cout;

int main() {
  cin.tie(nullptr)->sync_with_stdio(false);
  int T, n;
  solve();  // 预处理g数组
  cin >> T;
  for (int i = 1; i <= T; ++i) {
    cin >> n;
    cout << (g[n] + 1) * n / 2 << '\n';
  }
  return 0;
}
```

「BZOJ 2154」Crash 的数字表格

求值（对 20101009 取模）

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\operatorname{lcm}(i,j)\qquad(n,m\leqslant 10^7)$$

易知原式等价于

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{i\cdot j}{\gcd(i,j)}$$

枚举最大公因数 d，显然两个数除以 d 得到的数互质

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{d|i,d|j,\gcd(\frac{i}{d},\frac{j}{d})=1}\frac{i\cdot j}{d}$$

非常经典的 $\gcd$ 式子的化法

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

后半段式子中，出现了互质数对之积的和，为了让式子更简洁就把它拿出来单独计算。于是我们记

$$\mathrm{sum}(n,m)=\sum_{i=1}^{n}\sum_{j=1}^{m}[\gcd(i,j)=1]\ i\cdot j$$

接下来对 $\mathrm{sum}(n,m)$ 进行化简。首先枚举约数，并将 $[\gcd(i,j)=1]$ 表示为 $\varepsilon(\gcd(i,j))$

$$\sum_{d=1}^{n}\sum_{d|i}^{n}\sum_{d|j}^{m}\mu(d)\cdot i\cdot j$$

设 $i=i'\cdot d$，$j=j'\cdot d$，显然式子可以变为

$$\sum_{d=1}^{n}\mu(d)\cdot d^2\cdot\sum_{i=1}^{\lfloor\frac{n}{d}\rfloor}\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}i\cdot j$$

观察上式，前半段可以预处理前缀和；后半段又是一个范围内数对之和，记

$$g(n,m)=\sum_{i=1}^{n}\sum_{j=1}^{m}i\cdot j=\frac{n\cdot(n+1)}{2}\times\frac{m\cdot(m+1)}{2}$$

可以 $\Theta(1)$ 求解

至此

$$\mathrm{sum}(n,m)=\sum_{d=1}^{n}\mu(d)\cdot d^2\cdot g(\lfloor\frac{n}{d}\rfloor,\lfloor\frac{m}{d}\rfloor)$$

我们可以 $\lfloor\frac{n}{\lfloor\frac{n}{d}\rfloor}\rfloor$ 数论分块求解 $\mathrm{sum}(n,m)$ 函数。

在求出 $\mathrm{sum}(n,m)$ 后，回到定义 sum 的地方，可得原式为

$$\sum_{d=1}^{n}d\cdot\mathrm{sum}(\lfloor\frac{n}{d}\rfloor,\lfloor\frac{m}{d}\rfloor)$$

可见这又是一个可以数论分块求解的式子！

本题除了推式子比较复杂、代码细节较多之外，是一道很好的莫比乌斯反演练习题！（上述过程中，默认 $n \leqslant m$）

时间复杂度：$\Theta(n+m)$（瓶颈为线性筛）

代码实现

```cpp
#include <algorithm>
#include <iostream>
using namespace std;

constexpr int N = 1e7;
constexpr int mod = 20101009;
int n, m, mu[N + 5], p[N / 10 + 5], sum[N + 5];
bool flg[N + 5];

int Sum(int x, int y) {
  return ((long long)1 * x * (x + 1) / 2 % mod) *
         ((long long)1 * y * (y + 1) / 2 % mod) % mod;
}

int func(int x, int y) {
  int res = 0;
  int j;
  for (int i = 1; i <= min(x, y); i = j + 1) {
    j = min(x / (x / i), y / (y / i));
    res = (res + (long long)1 * (sum[j] - sum[i - 1] + mod) *
                      Sum(x / i, y / i) % mod) %
          mod;  //+mod防负数
  }
  return res;
}

int solve(int x, int y) {
  int res = 0;
  int j;
  for (int i = 1; i <= min(x, y); i = j + 1) {  // 整除分块处理
    j = min(x / (x / i), y / (y / i));
    res = (res + (long long)1 * (j - i + 1) * (i + j) / 2 % mod *
                      func(x / i, y / i) % mod) %
          mod;  // ! 每步取模防爆
  }
  return res;
}

void init() {  // 线性筛
  mu[1] = 1;
  int tot = 0, k = min(n, m);
  for (int i = 2; i <= k; ++i) {
    if (!flg[i]) {
      p[++tot] = i;
      mu[i] = -1;
    }
    for (int j = 1; j <= tot && i * p[j] <= k; ++j) {
      flg[i * p[j]] = true;
      if (i % p[j] == 0) {
```

```
        mu[i * p[j]] = 0;
        break;
      }
      mu[i * p[j]] = -mu[i];
    }
  }
  for (int i = 1; i <= k; ++i)
    sum[i] = (sum[i - 1] + (long long)1 * i * i % mod * (mu[i] +
mod)) % mod;
}

int main() {
  cin >> n >> m;
  init();
  cout << solve(n, m) << '\n';
}
```

「SDOI2015」约数个数和

多组数据，求

$$\sum_{i=1}^{n}\sum_{j=1}^{m}d(i\cdot j)\qquad\left(n,m,T\leq 5\times 10^4\right)$$

其中 $d(n)=\sum_{i|n}1$，$d(n)$ 表示 n 的约数个数

要推这道题首先要了解 d 函数的一个特殊性质

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

再化一下这个式子

$$\begin{aligned}
d(i\cdot j)&=\sum_{x|i}\sum_{y|j}[\gcd(x,y)=1]\\
&=\sum_{x|i}\sum_{y|j}\sum_{p|\gcd(x,y)}\mu(p)\\
&=\sum_{p=1}^{min(i,j)}\sum_{x|i}\sum_{y|j}[p\mid\gcd(x,y)]\cdot\mu(p)\\
&=\sum_{p|i,p|j}\mu(p)\sum_{x|i}\sum_{y|j}[p\mid\gcd(x,y)]\\
&=\sum_{p|i,p|j}\mu(p)\sum_{x|\frac{i}{p}}\sum_{y|\frac{j}{p}}1\\
&=\sum_{p|i,p|j}\mu(p)d\left(\frac{i}{p}\right)d\left(\frac{j}{p}\right)
\end{aligned}$$

将上述式子代回原式

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{m} d(i \cdot j) \\
=&\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{p \mid i, p \mid j} \mu(p) d\left(\frac{i}{p}\right) d\left(\frac{j}{p}\right) \\
=&\sum_{p=1}^{min(n,m)}\sum_{i=1}^{n}\sum_{j=1}^{m} [p \mid i, p \mid j] \cdot \mu(p) d\left(\frac{i}{p}\right) d\left(\frac{j}{p}\right) \\
=&\sum_{p=1}^{min(n,m)}\sum_{i=1}^{\left\lfloor \frac{n}{p} \right\rfloor}\sum_{j=1}^{\left\lfloor \frac{m}{p} \right\rfloor} \mu(p) d(i) d(j) \\
=&\sum_{p=1}^{min(n,m)} \mu(p) \sum_{i=1}^{\left\lfloor \frac{n}{p} \right\rfloor} d(i) \sum_{j=1}^{\left\lfloor \frac{m}{p} \right\rfloor} d(j) \\
=&\sum_{p=1}^{min(n,m)} \mu(p) S\left(\left\lfloor \frac{n}{p} \right\rfloor\right) S\left(\left\lfloor \frac{m}{p} \right\rfloor\right) \left(S(n) = \sum_{i=1}^{n} d(i)\right)
\end{aligned}
$$

那么 $O(n)$ 预处理 μ, d 的前缀和，$O(\sqrt{n})$ 分块处理询问，总复杂度 $O(n + T\sqrt{n})$.

代码实现

```cpp
#include <algorithm>
#include <iostream>
using namespace std;
constexpr long long N = 5e4 + 5;
long long n, m, T, pr[N], mu[N], d[N], t[N],
    cnt;  // t 表示 i 的最小质因子出现的次数
bool bp[N];

void prime_work(long long k) {
  bp[0] = bp[1] = true, mu[1] = 1, d[1] = 1;
  for (long long i = 2; i <= k; i++) {  // 线性筛
    if (!bp[i]) pr[++cnt] = i, mu[i] = -1, d[i] = 2, t[i] = 1;
    for (long long j = 1; j <= cnt && i * pr[j] <= k; j++) {
      bp[i * pr[j]] = true;
      if (i % pr[j] == 0) {
        mu[i * pr[j]] = 0;
        d[i * pr[j]] = d[i] / (t[i] + 1) * (t[i] + 2);
        t[i * pr[j]] = t[i] + 1;
        break;
      } else {
        mu[i * pr[j]] = -mu[i];
        d[i * pr[j]] = d[i] << 1;
        t[i * pr[j]] = 1;
      }
    }
  }
  for (long long i = 2; i <= k; i++)
    mu[i] += mu[i - 1], d[i] += d[i - 1];  // 求前缀和
}

long long solve() {
  long long res = 0, mxi = min(n, m);
  for (long long i = 1, j; i <= mxi; i = j + 1) {  // 整除分块
    j = min(n / (n / i), m / (m / i));
    res += d[n / i] * d[m / i] * (mu[j] - mu[i - 1]);
  }
  return res;
}

int main() {
  cin.tie(nullptr)->sync_with_stdio(false);
  cin >> T;
  prime_work(50000);  // 预处理
  while (T--) {
    cin >> n >> m;
    cout << solve() << '\n';
  }
  return 0;
}
```

莫比乌斯反演扩展

结尾补充一个莫比乌斯反演的非卷积形式。

对于数论函数 f,g 和完全积性函数 t 且 $t(1)=1$：

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

我们证明一下

$$\begin{aligned}
g(n)&=\sum_{i=1}^{n}\mu(i)t(i)f\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\
&=\sum_{i=1}^{n}\mu(i)t(i)\sum_{j=1}^{\left\lfloor\frac{n}{i}\right\rfloor}t(j)g\left(\left\lfloor\frac{\left\lfloor\frac{n}{i}\right\rfloor}{j}\right\rfloor\right)\\
&=\sum_{i=1}^{n}\mu(i)t(i)\sum_{j=1}^{\left\lfloor\frac{n}{i}\right\rfloor}t(j)g\left(\left\lfloor\frac{n}{ij}\right\rfloor\right)\\
&=\sum_{T=1}^{n}\sum_{i=1}^{n}\mu(i)t(i)\sum_{j=1}^{\left\lfloor\frac{n}{i}\right\rfloor}[ij=T]t(j)g\left(\left\lfloor\frac{n}{T}\right\rfloor\right) &\text{【先枚举 ij 乘积】}\\
&=\sum_{T=1}^{n}\sum_{i|T}\mu(i)t(i)t\left(\frac{T}{i}\right)g\left(\left\lfloor\frac{n}{T}\right\rfloor\right) &\text{【}\sum_{j=1}^{\left\lfloor\frac{n}{i}\right\rfloor}[ij=T]\text{对答案的贡献为 1，于是省略】}\\
&=\sum_{T=1}^{n}g\left(\left\lfloor\frac{n}{T}\right\rfloor\right)\sum_{i|T}\mu(i)t(i)t\left(\frac{T}{i}\right)\\
&=\sum_{T=1}^{n}g\left(\left\lfloor\frac{n}{T}\right\rfloor\right)\sum_{i|T}\mu(i)t(T) &\text{【t 是完全积性函数】}\\
&=\sum_{T=1}^{n}g\left(\left\lfloor\frac{n}{T}\right\rfloor\right)t(T)\sum_{i|T}\mu(i)\\
&=\sum_{T=1}^{n}g\left(\left\lfloor\frac{n}{T}\right\rfloor\right)t(T)\varepsilon(T) &\text{【}\mu*1=\varepsilon\text{】}\\
&=g(n)t(1) &\text{【当且仅当 T=1,}\varepsilon(T)=1\text{时】}\\
&=g(n) &\square
\end{aligned}$$

参考文献

algocode 算法博客

本页面贡献者：Ir1d, StudyingFather, Enter-tainer, mgt, ShaoChenHeng, H-J-Granger, Marcythm, orzAtalod, Siyuan, sshwy, Early0v0, Peanut-Tang, Xeonacid, countercurrent-time, ezoixx130, hyp1231, NachtgeistW, ranwen, GekkaSaori, ksyx, MegaOwIer, SamZhangQingChuan, Vxlimo, 383494, AngelKitty, CCXXXI, cjsoft, diauweb, Gesrua, Great-designer, guodong2005, Henry-ZHR, iamtwz, Konano, Lcyanstars, LovelyBuggies, Luckyblock233, Makkiy, Menci, minghu6, mxr612, ouuan, P-Y-Y, PotassiumWings, Suyun51 Tiphereth-A, weiyong1024, Chrogeek, CyaceQuious, FFjet, frank-xjh, GavinZhengOI, hehelego, HeRaNO, hjsjhn, hydingsy, i-yyi, kenlig, kxccc, luojiny1, lychees, nalemy, qwqAutomaton, shawlleyw, Sshwy, SukkaW, UserUnauthorized, WineChord, yjl9903