数论函数

一、定义

数论函数：定义域为正整数的函数。若无特殊说明下文讨论的函数均为数论函数。

积性函数：对于任意两个互质的正整数 a,b，均满足 $f(ab)=f(a)f(b)$ 的函数 f

完全积性函数：对于所有正整数 a,b，均满足 $f(ab)=f(a)f(b)$ 的函数 f

容易看出对于任意积性函数，均有 $f(1)=1$.

考虑一个大于 1 的正整数 n，设 $n=\prod p_i^{a_i}$，其中 p_i 为互不相同的质数。

那么对于一个积性函数 f，$f(n)=f(\prod p_i^{a_i})=\prod f(p_i^{a_i})$，若 f 还满足完全积性，则 $f(n)=\prod f(p_i)^{a_i}$

二、常见积性函数

欧拉函数 $\varphi(n)$：$\varphi(n)$ 表示 $[1,n]$ 中与 n 互质的整数的个数。结合中国剩余定理，容易证明 $\varphi(n)$ 是积性函数，但不是完全积性函数。

考虑一个质数 p 与一个正整数 k，由定义不难得出 $\varphi(p^k)=p^k-p^{k-1}=(p-1)p^{k-1}$，结合 $\varphi(n)$ 是积性函数，即可得到若 $n=\prod p_i^{a_i}$，则 $\varphi(n)=\prod (p_i-1)p_i^{a_i-1}$.

也可以写成容斥的形式：$\varphi(n)=n\cdot\prod_{p\in P}\frac{p-1}{p}$，其中 P 是 n 的不同质因子集合。

欧拉函数的一个小性质：$n>1$ 时，$[1,n]$ 中与 n 互质的整数的和为 $\frac{n\cdot\varphi(n)}{2}$.

莫比乌斯函数 $\mu(n)$：若 n 有平方数因子，则 $\mu(n)=0$。否则若 n 为 k 个不同质数之积，则 $\mu(n)=(-1)^k$。

$$\mu(n)=\begin{cases}1 & n=1\\ (-1)^k & n=p_1p_2\cdots p_k\\ 0 & else\end{cases}$$

除数函数：$\sigma_k(n)$表示n的所有正因子的k次幂之和。

$d(n)=\sigma_0(n)$表示n的正因子个数，$\sigma(n)=\sigma_1(n)$表示n的所有正因子之和。

幂函数：$Id_k(n)=n^k$，$1(n)=Id_0(n)=1$，$Id(n)=Id_1(n)=n$

单位函数：$e(n)=\varepsilon(n)=\begin{cases}1 & n=1\\ 0 & n>1\end{cases}$

三、狄利克雷（Dirichlet）卷积

定义两个数论函数f,g的 Dirichlet 卷积：

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

狄利克雷卷积的性质：

交换律：$f*g=g*f$
结合律：$(f*g)*h=f*(g*h)$
分配律：$f*(g+h)=f*g+f*h$
单位元：$f*\varepsilon=\varepsilon*f=f$
若f,g均为积性函数，则$f*g$也为积性函数

常见的狄利克雷卷积：

$$d(n)=\sum_{d|n}1 \quad 即d=1*1$$

$$\sigma(n)=\sum_{d|n}d \quad 即\sigma=Id*1$$

$$\varphi(n)=\sum_{d|n}\mu(d)\frac{n}{d} \quad 即\varphi=\mu*Id$$

$$\varepsilon(n)=\sum_{d|n}\mu(d) \quad 即\varepsilon=\mu*1$$

$\varphi = \mu * Id, \varepsilon = \mu * 1 \Rightarrow$

由于 $1 * \varphi = 1 * \mu * Id \Rightarrow$ $Id = \varphi * 1$ ，即 $n = \sum_{d|n} \varphi(d)$

由 $\varepsilon = \mu * 1$ 也可知：$\sum_{d|n} \mu(d) = [n = 1]$

$[e]$ 指的是，若 e 这个表达式为真那么 $[e] = 1$ 否则 $[e] = 0$

证明：设 n 有 $k(k > 0)$ 个不同质因子。

那么，n 所有因子中 μ 不为 0 的只有所有质因子次数都为 1 的因子，质因子个数为 i 的因子有 $C(k,i)$ 个，即：

$$\begin{aligned}\sum_{d|n} \mu(d) &= \sum_{i=0}^{k} (-1)^i \cdot C(k,i) \\ &= (1-1)^k \\ &= 0\end{aligned}$$

$n = 1$ 时式子只有一项和即为 1，因此 $\sum_{d|n} \mu(d) = [n = 1]$ 得证。

四、莫比乌斯反演：

若有两个函数 f, g 满足：

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

则它们也满足

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

反之亦然，即

$$f = g * 1 \Leftrightarrow g = \mu * f$$

通过几个常见卷积容易证明这个反演公式。（只需在左边等式的等号两边同卷上 μ ）

还可以通过下面这个等式来证明：

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

$$\begin{aligned}g(n)&=\sum_{m|n}\left[\frac{n}{m}=1\right]g(m)\\&=\sum_{m|n}\sum_{d|\frac{n}{m}}\mu(d)g(m)\\&=\sum_{d|n}\sum_{m|\frac{n}{d}}\mu(d)g(m)\\&=\sum_{d|n}\mu(d)\sum_{m|\frac{n}{d}}g(m)\\&=\sum_{d|n}\mu(d)f(\frac{n}{d})\end{aligned}$$

它的另一个形式：

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

证明：

$$\begin{aligned}f(n)&=\sum_{n|d}g(d)\\g(n)&=\sum_{n|d}\mu(\frac{d}{n})f(d)\\&=\sum_{n|d}\mu(\frac{d}{n})\sum_{d|t}g(t)\\&=\sum_{n|t}g(t)\sum_{d|\frac{t}{n}}\mu(d)\\&=\sum_{n|t}g(t)[\frac{t}{n}=1]\\&=\sum_{n|t}g(t)[t=n]\\&=g(n)\end{aligned}$$

五、莫比乌斯反演与容斥原理

由上面常见的狄利克雷卷积式我们可以得到 $\varphi(n)=\sum_{d|n}\frac{\mu(d)\cdot n}{d}$

我们考虑用容斥原理的计算方式来理解这个式子：

考虑不超过 n 的正整数 k，根据欧拉函数定义，我们要算出有多少个 k 与 n 互质。

令 $d=\gcd(n,k)$，当 $d>1$ 时，所有这样的 k 要被去除。考虑质数集合 $P=\{2,3,5,7,\cdots\}$，当 $d>1$ 时显然 k 会是 P 中某些质数的倍数。若 $p\mid d$，可知满足这样条件的 k 有 $\frac{n}{p}$ 个，这些数都要被去掉，但很容易发现这样做是会重复计数的。

继续考虑两个不同质数 p_1,p_2，若 $p_1p_2\mid d$，则这样的 d 在上一步会被重复去掉两次，所以需要加上 $\frac{n}{p_1p_2}$.

接着考虑三个不同质数 p_1,p_2,p_3 的情况，若 $p_1p_2p_3\mid d$ 则它在开始时被去掉三次，但是在考虑两个质数时又被加回三次，因此需要再次去掉 $\frac{n}{p_1p_2p_3}$.

这样的话考虑 t 个不同质数 $p_1,p_2,p_3\cdots p_t$，若 $p_1p_2p_3\cdots p_t\mid d$，根据容斥原理，答案需要加上 $(-1)^t\cdot\frac{n}{p_1p_2p_3\cdots p_t}$.

而通过莫比乌斯函数的定义以及上面的反演公式我们可以发现，d 其实就是若干不同质数的乘积，因此 $\mu(d)$ 的作用就是 $(-1)^t$。所以莫比乌斯反演的本质就是容斥。

六、线性筛

设 pr_i 表示 i 的最小质因子。

考虑从 2 到 n 枚举 i，若 pr_i 还未被计算出来，那么 i 为质数，$pr_i=i$。

枚举所有不超过 pr_i 的质数 p，使 $pr_{(i\cdot p)}=p$.

相当于将每个数的质因子降序分解，由于分解方式是唯一的，于是每个数只会被筛到一次，复杂度为 $O(n)$。并且在筛出质数的同时我们还能得到每个数的最小质因子，进一步地

我们可以得到每个数去除所有最小质因子后的值，这对于我们计算积性函数非常方便。

代码：

```
void sieve() {
  for (int i = 2; i < N; ++i) {
    if (!pr[i])
      prime[pn++] = pr[i] = i;
    for (int j = 0; j < pn; ++j) {
      if (i * prime[j] >= N) break;
      pr[i * prime[j]] = prime[j];
      if (i % prime[j] == 0) break;
      // 也可写成 if (pr[i] == prime[j]) break;
    }
  }
}
```

同样我们也可以这么理解它的复杂度：

设合数 n 最小的质因数为 p，它的另一个大于 p 的质因数为 p'，令 $n = pm = p'm'$。观察上面的程序片段，可以发现 j 循环到质因数 p 时合数 n 第一次被标记（若循环到 p 之前已经跳出循环，说明 n 有更小的质因数），若也被 p' 标记，则是在这之前（因为 $m' < m$），考虑 i 循环到 m'，注意到 $n = pm = p'm'$ 且 p, p' 为不同的质数，因此 $p \mid m'$，所以当 j 循环到质数 p 后结束，不会循环到 p'，这就说明不会被 p' 筛去。

利用线性筛求解每个数 n 的最小质因子 p 的次数 k：令 $n = pm$，由于 p 是 n 最小的质因子，若 $p^2 \mid n$，则 $p \mid m$，并且 p 也是 m 最小的质因子，这样在筛法的同时即可算出新筛去的合数的最小质因子的次数。

在线性筛中，我们能得到每个数 n 的最小质因数 p 以及它的次数 k，这样在求解积性函数时，我们能利用 $f(n) = f(p^k) f(\frac{n}{p^k})$，在线性筛的同时快速求出 $f(n)$。

但是这还不够，一般我们要能快速求解 $f(p^k)$，这时一般就要结合 f 函数的性质考虑。

例如求欧拉函数 φ，$\varphi(p^k) = (p-1)p^{k-1}$，因此筛法时，若 $p \mid m$，φ 就乘上 p，否则

乘上 $p-1$。（这里的 m 指的是 $n=pm$ 中的 m ）

再比如莫比乌斯函数 μ，只有当 $k=1$ 时 $\mu(p^k)=-1$，否则 $\mu(p^k)=0$，因此和欧拉函数一样根据是否有 $p\mid m$ 进行判断处理即可。

代码：

```
void sieve() {
  phi[1] = mu[1] = 1;
  for (int i = 2; i < N; ++i) {
    if (!pr[i])
      prime[pn++] = pr[i] = i, phi[i] = i - 1, mu[i] = -1;
    for (int j = 0; j < pn; ++j) {
      int k = i * prime[j];
      if (k >= N) break;
      pr[k] = prime[j];
      if (i % prime[j] == 0) {
        phi[k] = phi[i] * prime[j];
        mu[k] = 0;
        break;
      } else {
       phi[k] = phi[i] * (prime[j] - 1);
       mu[k] = -mu[i];
      }
    }
  }
}
```

模大质数 P 意义下 n 的乘法逆元 $f(n)$ 是一个完全积性函数，因此求解小于 n 的所有逆元 $f(1),f(2),\cdots,f(n)$，也能使用筛法求解。

但其实处理逆元我们有更简单的顺推方法：

设 $P=an+b$，即 $b=P \bmod n, a=\left\lfloor \frac{P}{n} \right\rfloor$，则有：

$$
\begin{aligned}
an+b &\equiv 0 (\bmod P) \\
b &\equiv -an (\bmod P) \\
n^{-1}b &\equiv (P-a) (\bmod P) \\
n^{-1} &\equiv (P-a)\cdot b^{-1} (\bmod P)
\end{aligned}
$$

七、应用

对于一些问题，我们要求函数 $f(n)$，如果我们很难直接求出 $f(n)$ 的值，但容易求出它的约数和或倍数和，那么我们可以通过莫比乌斯反演来求得 $f(n)$。

例如：求所有由小写字母组成的（不一定要使用所有字母），长度为 n 的字符串中，没有周期性的字符串的个数。如果一个长度为 n 的字符串可以通过一个长度为 $m(m<n)$ 的字符串重复 n/m 次后得到，我们就称它具有周期性。$n \le 10^9$。

设最小循环节长度恰好为 m 的字符串的个数为 $f(m)$，题目所求的即为 $f(n)$。

再设最小循环节长度为 m 的约数的字符串个数为 $g(m)$，则我们有 $g(m)=\sum_{d|m} f(d)$.

实际上 $g(m)=26^m$。因为显然把任意长度为 m 的字符串重复 n/m 次所得到的字符串，它的最小循环节长度至多为 m，且循环节长度一定为 m 的约数。

再根据莫比乌斯反演公式 $g(m)=\sum_{d|m} f(d) \Leftrightarrow f(m)=\sum_{d|m} g(d)\mu(\frac{m}{d})$，现在已经可以快速求得 $g(d)$，因此我们只要将 n 质因数分解，搜出它所有的因子并计算好因子们的 μ 值，利用这个反演公式我们即可在 $O(\sqrt{n})$ 的预处理时间、$O(\text{约数个数})$ 的计算时间内解决此题。（10^9 以内的数的约数个数不超过 1000）。

在竞赛中莫比乌斯反演还有一种应用就是用来大力计算数论函数求和问题。这在下面的例题以及练习题中将会体现。

八、例题

题目大意：

T 组询问，每次询问给定 n,m (假定 $n<m$)，问有多少个数对 (x,y) 满足 $\gcd(x,y)=1$ 且 $1 \le x \le n, 1 \le y \le m$。$n,m \le 10^6, T \le 10^3$.

分析：

首先我们利用求和符号以及 $[e]$ 符号将所求值表达出来：

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

接着我们利用之前提到的 $\sum_{d|n}\mu(d)=[n=1]$，将式子改写并继续推导：

$$\begin{aligned}ans&=\sum_{i=1}^{n}\sum_{j=1}^{m}[\gcd(i,j)=1]\\&=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{d|\gcd(i,j)}\mu(d)\\&=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{\substack{d|i\\d|j}}\mu(d)\\&=\sum_{d}\sum_{\substack{i=1\\d|i}}^{n}\sum_{\substack{j=1\\d|j}}^{m}\mu(d)\\&=\sum_{d}\mu(d)\cdot\left\lfloor\frac{n}{d}\right\rfloor\cdot\left\lfloor\frac{m}{d}\right\rfloor\end{aligned}$$

化简到这一步，预处理 μ 后已经可以在 $O(n)$ 时间内解决单组询问了。

下面考虑继续优化。

首先容易看出 $\left\lfloor\frac{n}{d}\right\rfloor$ 的取值只有 $2\lfloor\sqrt{n}\rfloor$ 种，同理 $\left\lfloor\frac{m}{d}\right\rfloor$ 的取值只有 $2\lfloor\sqrt{m}\rfloor$，并且相同的取值对应的 i 都是连续的一段区间，因此 $\left\lfloor\frac{n}{d}\right\rfloor$ 与 $\left\lfloor\frac{m}{d}\right\rfloor$ 都相同的区间只有 $2\lfloor\sqrt{n}\rfloor+2\lfloor\sqrt{m}\rfloor$ 个。因此在上式中我们需要枚举的 d 也就只有 $2\lfloor\sqrt{n}\rfloor+2\lfloor\sqrt{m}\rfloor$ 个了，注意此时我们需要预处理 μ 的前缀和来快速计算。

枚举除法的取值这种方法在莫比乌斯反演当中非常常用，且代码也很简单：

```
for (int i = 1; i <= n; ) {  // 单个 n
  int j = n / (n / i);
  ans += (n / i) * (sum[j] - sum[i - 1]);
  i = j + 1;
}

```

```
for (int i = 1; i <= n; ) {  // n与m
  int j = std::min(n / (n / i), m / (m / i));
  ans += (n / i) * (m / i) * (sum[j] - sum[i - 1]);
  i = j + 1;
}
```

利用这个技巧，此题即可在$O(T\sqrt{n})$时间内解决。

代码：

```
#include <algorithm>
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
using namespace std;

typedef long long ll;
const int N = 5e4 + 1;
int T_T, a, b, n, tot, nxt, pri[N], miu[N], sum[N];
bool check[N];
ll ans;

inline void Init() {
    miu[1] = 1; check[0] = check[1] = true;
    for (int i = 2; i < N; ++i) {
        if (!check[i])
            pri[++tot] = i, miu[i] = -1;
        for (int j = 1; j <= tot; ++j) {
            if ((ll)i * pri[j] >= N) break;
            check[i * pri[j]] = true;
            if (i % pri[j] == 0) break;
            else miu[i * pri[j]] = -miu[i];
        }
    }
    for (int i = 1; i < N; ++i)
        sum[i] = sum[i - 1] + miu[i];
    return ;
}

char ch;
inline int read() {
    while (ch = getchar(), ch < '0' || ch > '9');
    int res = ch - 48;
    while (ch = getchar(), ch >= '0' && ch <= '9') res = res * 10 + ch - 48;
```

```
    return res;
}

int main() {
    Init();
    T_T = read();
    while (T_T--) {
        ans = 0;
        a = read(); b = read();
        n = a < b ? a : b;
        for (int i = 1; i <= n;) {
            nxt = min(a / (a / i), b / (b / i));
            ans += (ll)(sum[nxt] - sum[i - 1]) * (a / i) * (b / i);
            i = nxt + 1;
        }
        printf("%lld\n", ans);
    }
    return 0;
}
```

练习题

BZOJ 1257 ; BZOJ 2705 ; BZOJ 2226 ; BZOJ 1101 ; BZOJ 2820 ;

BZOJ 2005 ; BZOJ 2693 ; BZOJ 2694

上述内容参考：

1.数论函数选讲，任之洲

2.线性筛法与积性函数，贾志鹏

3.挑战程序设计竞赛（第二版）