□□

krr 数论全笔记

□□

□□□□

- 整除

- 素数判定

- 线性筛

- 互质

- 约数

- 调和级数
- 余数

□□□□□□

每个数，都可以唯一被分解成 $p_1^{c_1}p_2^{c_2}...p_n^{c_n}$ 的形式，其中 p_i 是素数。

□□

a, b 互质，即没有公共的质因子。即在唯一分解后，所有的 p_i 不相同。

□□

通常有两种写法... 这份笔记中，所有的都依照后面一种。

若 $a\, mod\, b = 0$，则 a 能被 b 整除，或 b 能整除 a，写作 $a|b$
若 $b = ka$，记为 b 被 a 整除，a 整除于 b。写作 $a|b$。

绕一绕的话...$n = km$，读作 m 整除 n，n 被 m 整除，$m|n$，$|$ 的前后顺序依照 x 整除 y 的形式... 绕一绕... 过一会儿就绕好了 ~ 总之记住，小的放前面，大的放后面。

□□□□

若一个数 N 是合数，一定存在 $T \leq \sqrt{n}$，且 T 能整除 N。

反证法，假设命题不成立，那么一定存在 $T > \sqrt{n}$ 且 $T|N$。那么一定存在 $\frac{N}{T} \leq \sqrt{n}$ 且 $\frac{N}{T}|N$。则命题成立 ~
代码为试除法

```
inline bool prime(ll val)
{
  for (int i = 2; i <= sqrt(val); ++i)
    if(val % i == 0) return false;
  return true;
}
```

□□□

普通筛法略过了，我们讲讲线性筛。
线性筛 $O(n)$ 的原理是，使每个数只被自己最小的质因子筛一次。
都知道 $a \in N$，b 是质数，那么 $a \cdot b$ 一定不是质数。

我们令每个合数只被自己最小的质因子 p 筛出来，那么每个合数只会被筛一次。
我们对于每一个数 a，用 a 去乘上 $\leq a$ 的所有质数。
接下来给出，到每个数的时候，它一定被筛过的证明。
这里是我万年理解不到的... 别的教程也不提这个...so 郁闷
请把这一段认真看完... 我尽量保证了语言没问题。
我们的筛法是，标记某个小于其本身质数 $\times$ 某个小于其本身的合数。
那么，假设，在进行那个“小于其本身的素数”的处理的时候，我们枚举了所有小于其本身的合数，而那时的 $break$ 规则是 $prm[j] * val > n$，这个数一定是 $\leq n$ 的，所以其一定被筛过。这种反向思考很赞，但是别人为啥都不提这一点呢。但是啊，普通筛的时候，这么看吧！假如说当前的数可以被分解成 $p_1^{c_1}...$，那么它会被所有的 p 筛一次，而某个数的素因子的个数是 $\log n$ 级别的，那么复杂度 $n \log n$。线性筛保证每个数一定只会被其最小的质因子筛过一次，根据上面的描述，已经足够了，所以线性筛的复杂度是 $O(n)$。
代码如下

```
inline void prime()
{
  for (int i = 2; i <= n; ++i)
  {
    if(!vis[i]) prm[++cnt] = i;
    for (int j = 1; j <= cnt; ++j)
    {
      if(i * prm[j] > n) break;
      vis[i * prm[j]] = 1;
      if(i % prm[j] == 0) break;
    }
  }
}
```

代码的顺序和刚才思维的顺序略有不同，而这正好是线性筛的妙处啊！

尝试一下，并不好实现的思维题
>>> [ep1] 质因数分解 $N!$。
这样看 ~
利用线性筛的思路，我们每个合数只被其最小质因子筛一次。

我们知道一个数的质因数分解形式后，用这个数乘上一个质数，能够知道计算结果的质因数分解形式。我们对结果的形式累乘，就是答案 ~ 不优化的话，时间复杂度和空间复杂度都是 $O(n\log n)$，实现也会很复杂。仅练习思维。
>>> [ep2] 求 $[L,R]$ 的质数个数。$L,R\le 2^{31},R-L\le 10^6$。

可以筛 $R-L$ 的，那么我们先把 $[2,\sqrt{R}]$ 之间的数筛出来，这里的数一定可以组合出 $[L,R]$ 的所有数。然后对于所有质数，我们进行 $[L,R]$ 的标记。$vis[i\times p]=1,i\in[\frac{L}{p},\frac{R}{p}]$。

□□□□

并不能给出详细的解答，总之记住公式，$\sum_{i=1}^{n} \frac{n}{i} = n \ln(n+1) + nr$。$r$ 为欧拉常数，约等于 0.5772156649，所以遇到形如 $\sum_{i=1}^{n} \frac{n}{i}$ 的算式，在时间复杂度中通常记作 $n \log n$，或 $n \ln n$。我习惯前者。调和级数通常用作时间复杂度的证明。

□□

下面介绍一点约数常用定理。
定义 N。
约数的定义为 $d|N$，d 为约数。

可以简单发现，p_i 是 N 的约数，也就是说，$p_1^{c_1} p_2^{c_2} ... p_n^{c_n}$ 的子集都是其一个约数。
假设 N 进行唯一分解后的数是 $p_1^{c_1} p_2^{c_2} ... p_n^{c_n}$。
N 的正约数的个数 $= \prod_{i=1}^{m} (c_i + 1)$。乘法原理

N 的正约数和为 $\prod_{i=1}^{m} (\sum_{j=0}^{c_i} (p_i)^j)$。手玩 ~
N 的正整数约数集合是试除法，这里不再赘述...
还有一个求法。

对于数字 d，d 一定是 $d \cdot k$ 的约数。这样筛下来，复杂度是 $O(\sum_{i=1}^{n} \frac{n}{i})$。利用调和级数，$O(n \log n)$。比 $O(n\sqrt{n})$ 要好得多。

□□

$a\ mod\ b = c$，称 c 是 a 除以 b 的余数。余数的定义式如下

$$a \; mod \; b = a - \lfloor \frac{a}{b} \rfloor \cdot b$$

看一个例题：$BZOJ1257$。求 $\sum_{i=1}^{n} k \; mod \; i$。

转化题目，求 $\sum_{i=1}^{n} (k - \lfloor \frac{k}{i} \rfloor \cdot i) \Rightarrow k \times n - \sum_{i=1}^{n} (\lfloor \frac{k}{i} \rfloor \cdot i)$。由于复杂度的问题，我们从 $\frac{k}{i}$ 入手，因为会有一段区间，使得其不变。我们设 $\lfloor \frac{k}{i} \rfloor = x$。这段区间的值是 $\frac{(i_{first}+i_{end}) \times (end-first+1)}{2} \times x$。所以问题来了，我们需要快速获取什么区间内，$\lfloor \frac{k}{i} \rfloor$ 不变。

结论如下，我也不知道怎么推的。$i \in [x, \lfloor \frac{k}{\lfloor \frac{k}{x} \rfloor} \rfloor]$ 时，$\lfloor \frac{k}{i} \rfloor$ 不变。x 是上一个的右端点 $+ 1$。具体请见代码，lyd 的代码如下：

```
#define ll long long

#define R register

#include <bits/stdc++.h>



using namespace std;



ll n, k, ans;

int main()

{
```

```
  scanf("%lld%lld", &n, &k);
  ans = n * k;
  for (int l = 1, r; l <= n; l = r + 1)
  {
    r = k / l ? min(k / (k / l), n) : n;
    ans -= (k / l) * (l + r) * (r - l + 1) / 2;
  }
  printf("%lld", ans);
}
```