数论与组合数学基础

福建省福州第一中学 钟知闲

February 8, 2018

目录

数论基础

- 整除：a整除b即$\frac{b}{a} \in \mathbf{Z}$，记作$a|b$
- 因数与倍数：$a|b$即a是b的因数，b是a的倍数
- 带余除法：对于整数a, b（$b \neq 0$），设a除以b的商为q，余数为r，则$a = bq + r$，q, r为整数且$0 \leq r < |b|$
- 模：a除以b余数为r，记为$a \bmod b = r$
- 同余：a, b模p同余即a, b除以p的余数相同，记作$a \equiv b \pmod{p}$

数论基础

- 整除：a整除b即$\frac{b}{a} \in \mathbf{Z}$，记作$a|b$
- 因数与倍数：$a|b$即a是b的因数，b是a的倍数
- 带余除法：对于整数a, b（$b \neq 0$），设a除以b的商为q，余数为r，则$a = bq + r$，q, r为整数且$0 \leq r < |b|$
- 模：a除以b余数为r，记为$a \bmod b = r$
- 同余：a, b模p同余即a, b除以p的余数相同，记作$a \equiv b \pmod p$

数论基础

- 整除：a整除b即$\frac{b}{a} \in \mathbf{Z}$，记作$a|b$
- 因数与倍数：$a|b$即a是b的因数，b是a的倍数
- 带余除法：对于整数a, b（$b \neq 0$），设a除以b的商为q，余数为r，则$a = bq + r$，q, r为整数且$0 \leq r < |b|$
- 模：a除以b余数为r，记为$a \bmod b = r$
- 同余：a, b模p同余即a, b除以p的余数相同，记作$a \equiv b \pmod p$

数论基础

- 整除：a整除b即$\frac{b}{a} \in \mathbf{Z}$，记作$a|b$
- 因数与倍数：$a|b$即a是b的因数，b是a的倍数
- 带余除法：对于整数a, b（$b \neq 0$），设a除以b的商为q，余数为r，则$a = bq + r$，q, r为整数且$0 \leq r < |b|$
- 模：a除以b余数为r，记为$a \bmod b = r$
- 同余：a, b模p同余即a, b除以p的余数相同，记作$a \equiv b \pmod{p}$

数论基础

- 整除：a整除b即$\frac{b}{a} \in \mathbf{Z}$，记作$a|b$
- 因数与倍数：$a|b$即a是b的因数，b是a的倍数
- 带余除法：对于整数a, b（$b \neq 0$），设a除以b的商为q，余数为r，则$a = bq + r$，q, r为整数且$0 \leq r < |b|$
- 模：a除以b余数为r，记为$a \bmod b = r$
- 同余：a, b模p同余即a, b除以p的余数相同，记作$a \equiv b \pmod{p}$

余数的性质

类比十进制运算个位数的规律，不难发现

- $(a \bmod p \pm b \bmod p) \bmod p = (a \pm b) \bmod p$
- $(a \bmod p)(b \bmod p) \bmod p = ab \bmod p$

注意C++的取模和取余是不一样的，C++的取模是保留符号的，如$(-4) \bmod 3 = 2$，但C++中(-4)%3=-1

余数的性质

类比十进制运算个位数的规律，不难发现

- $(a \bmod p \pm b \bmod p) \bmod p = (a \pm b) \bmod p$
- $(a \bmod p)(b \bmod p) \bmod p = ab \bmod p$

注意C++的取模和取余是不一样的，C++的取模是保留符号的，如$(-4) \bmod 3 = 2$，但C++中(-4)%3=-1

余数的性质

类比十进制运算个位数的规律，不难发现

- $(a \bmod p \pm b \bmod p) \bmod p = (a \pm b) \bmod p$
- $(a \bmod p)(b \bmod p) \bmod p = ab \bmod p$

注意C++的取模和取余是不一样的，C++的取模是保留符号的，如$(-4) \bmod 3 = 2$，但C++中(-4)%3=-1

余数的性质

类比十进制运算个位数的规律，不难发现

- $(a \bmod p \pm b \bmod p) \bmod p = (a \pm b) \bmod p$
- $(a \bmod p)(b \bmod p) \bmod p = ab \bmod p$

注意C++的取模和取余是不一样的，C++的取模是保留符号的，如$(-4) \bmod 3 = 2$，但C++中(-4)%3=-1

基本概念

- 质数：正因数只包含1和它本身的正整数（$2, 3, 5, 7 \cdots$）
- 合数：正因数包含除1和它本身外的数的正整数（$4, 6, 8, 9 \cdots$）

基本概念

- 质数：正因数只包含1和它本身的正整数（$2, 3, 5, 7 \cdots$）
- 合数：正因数包含除1和它本身外的数的正整数（$4, 6, 8, 9 \cdots$）

质数筛法

给定正整数$n \le 10^7$，求$[1, n]$内的所有质数

质数筛法

枚举$i = 1, 2, \cdots, n$，判断i是否包含$[2, \sqrt{i}]$内的因数（$x|i \Leftrightarrow \frac{i}{x}|i$，即因数集合具有“对称”性）

效率太低，考虑改进

质数筛法

枚举$i=1,2,\cdots,n$，判断i是否包含$[2,\sqrt{i}]$内的因数
（$x|i \Leftrightarrow \frac{i}{x}|i$，即因数集合具有“对称”性）
效率太低，考虑改进

质数筛法

枚举正整数$a, b \geq 2$且$ab \leq n$，将ab标记为合数

```
for(int a=2;a<=n;a++)
    for(int b=2;a*b<=n;b++)com[a*b]=1;
```

复杂度?

$$O(n + \frac{n}{2} + \frac{n}{3} + \cdots) = O(n \log n)$$

质数筛法

枚举正整数$a, b \geq 2$且$ab \leq n$，将ab标记为合数

```
for(int a=2;a<=n;a++)
    for(int b=2;a*b<=n;b++)com[a*b]=1;
```

复杂度?

$$O(n + \frac{n}{2} + \frac{n}{3} + \cdots) = O(n \log n)$$

质数筛法

优化：只要枚举a为质数的ab

埃拉托斯特尼筛法（埃氏筛法）：

```
for(int a=2;a<=n;a++)if(!com[a])
    for(int b=2;a*b<=n;b++)com[a*b]=1;
```

复杂度?

$$O(\frac{n}{2}+\frac{n}{3}+\frac{n}{5}+\cdots)=O(n\log\log n)$$

能做到$O(n)$吗?

质数筛法

优化：只要枚举a为质数的ab

埃拉托斯特尼筛法（埃氏筛法）：

```
for(int a=2;a<=n;a++)if(!com[a])
    for(int b=2;a*b<=n;b++)com[a*b]=1;
```

复杂度?

$$O(\frac{n}{2}+\frac{n}{3}+\frac{n}{5}+\cdots)=O(n\log\log n)$$

能做到$O(n)$吗?

质数筛法

优化：只要枚举a为质数的ab
埃拉托斯特尼筛法（埃氏筛法）：

```
for(int a=2;a<=n;a++)if(!com[a])
    for(int b=2;a*b<=n;b++)com[a*b]=1;
```

复杂度?

$$O(\frac{n}{2}+\frac{n}{3}+\frac{n}{5}+\cdots)=O(n\log\log n)$$

能做到$O(n)$吗?

质数筛法

线性筛法基本思想：每个数只被最小的质因子筛一次，即对于a是质数，b的最小质因子不小于a的整数对a, b，标记ab为合数

实现：先枚举b，再枚举a，枚举到$a|b$时结束

```
for(int i=2;i<=n;i++){
    if(!com[i])p[cnt++]=i;
    for(int j=0;j<cnt&&i*p[j]<=n;i++){
        com[i*p[j]]=1;
        if(i%p[j]==0)break;
    }
}
```

质数筛法

质数分布定理：$\pi(n) \sim \frac{n}{\ln n}$，$\pi(n)$为$n$以内质数个数

所以存放质数的数组可以开小一些

质数筛法

质数分布定理：$\pi(n) \sim \frac{n}{\ln n}$，$\pi(n)$为$n$以内质数个数
所以存放质数的数组可以开小一些

质因数分解

每个正整数n存在质因数分解形式$n=\prod_{i=1}^{m} p_i^{c_i} = p_1^{c_1} p_2^{c_2} \cdots p_m^{c_m}$，并且这种分解是唯一的

例：$7=7$，$18=2\times 3^2$......

给定正整数$n\le 10^{12}$，求n的质因数分解式

- 从1到n枚举因数：$O(n)$
- 从1到$\sqrt{n}$枚举因数：$O(\sqrt{n})$
- 线性预处理1到$\sqrt{n}$的质数，只枚举质数：预处理$O(\sqrt{n})$，单次分解$O(\frac{\sqrt{n}}{\log n})$

质因数分解

每个正整数n存在质因数分解形式$n=\prod_{i=1}^{m} p_i^{c_i}=p_1^{c_1}p_2^{c_2}\cdots p_m^{c_m}$，并且这种分解是唯一的

例：$7=7$，$18=2\times 3^2$......

给定正整数$n\le 10^{12}$，求n的质因数分解式

- 从1到n枚举因数：$O(n)$
- 从1到$\sqrt{n}$枚举因数：$O(\sqrt{n})$
- 线性预处理1到$\sqrt{n}$的质数，只枚举质数：预处理$O(\sqrt{n})$，单次分解$O(\frac{\sqrt{n}}{\log n})$

质因数分解

每个正整数n存在质因数分解形式$n=\prod_{i=1}^{m}p_i^{c_i}=p_1^{c_1}p_2^{c_2}\cdots p_m^{c_m}$，并且这种分解是唯一的

例：$7=7$，$18=2\times 3^2$......

给定正整数$n\le 10^{12}$，求n的质因数分解式

- 从1到n枚举因数：$O(n)$
- 从1到$\sqrt{n}$枚举因数：$O(\sqrt{n})$
- 线性预处理1到$\sqrt{n}$的质数，只枚举质数：预处理$O(\sqrt{n})$，单次分解$O(\frac{\sqrt{n}}{\log n})$

质因数分解

每个正整数n存在质因数分解形式$n = \prod_{i=1}^{m} p_i^{c_i} = p_1^{c_1} p_2^{c_2} \cdots p_m^{c_m}$，并且这种分解是唯一的

例：$7 = 7$，$18 = 2 \times 3^2 \ldots\ldots$

给定正整数$n \le 10^{12}$，求n的质因数分解式

- 从1到n枚举因数：$O(n)$
- 从1到$\sqrt{n}$枚举因数：$O(\sqrt{n})$
- 线性预处理1到$\sqrt{n}$的质数，只枚举质数：预处理$O(\sqrt{n})$，单次分解$O(\frac{\sqrt{n}}{\log n})$

质因数分解

每个正整数n存在质因数分解形式$n=\prod_{i=1}^{m} p_i^{c_i}=p_1^{c_1}p_2^{c_2}\cdots p_m^{c_m}$，并且这种分解是唯一的

例：$7=7$，$18=2\times 3^2$......

给定正整数$n\le 10^{12}$，求n的质因数分解式

- 从1到n枚举因数：$O(n)$
- 从1到$\sqrt{n}$枚举因数：$O(\sqrt{n})$
- 线性预处理1到$\sqrt{n}$的质数，只枚举质数：预处理$O(\sqrt{n})$，单次分解$O(\frac{\sqrt{n}}{\log n})$

质因数分解

每个正整数n存在质因数分解形式$n=\prod_{i=1}^{m} p_i^{c_i}=p_1^{c_1}p_2^{c_2}\cdots p_m^{c_m}$，并且这种分解是唯一的

例：$7=7$，$18=2\times 3^2$......

给定正整数$n\le 10^{12}$，求n的质因数分解式

- 从1到n枚举因数：$O(n)$
- 从1到$\sqrt{n}$枚举因数：$O(\sqrt{n})$
- 线性预处理1到$\sqrt{n}$的质数，只枚举质数：预处理$O(\sqrt{n})$，单次分解$O(\frac{\sqrt{n}}{\log n})$

质因数分解

每个正整数n存在质因数分解形式$n=\prod_{i=1}^{m} p_i^{c_i}=p_1^{c_1}p_2^{c_2}\cdots p_m^{c_m}$，并且这种分解是唯一的
例：$7=7$，$18=2\times 3^2$......
给定正整数$n\le 10^{12}$，求n的质因数分解式

- 从1到n枚举因数：$O(n)$
- 从1到$\sqrt{n}$枚举因数：$O(\sqrt{n})$
- 线性预处理1到$\sqrt{n}$的质数，只枚举质数：预处理$O(\sqrt{n})$，单次分解$O(\frac{\sqrt{n}}{\log n})$

质因数分解

给定正整数$n \le 10^7$，求n的质因数分解式

- $O(n)$预处理n以内质数，同时预处理每个合数x的最小质因子p_x
- 每次分解不断将n提取一个p_x
- 由于每提取一个质因子n就会减半，单次分解复杂度$O(\log n)$

质因数分解

给定正整数$n \le 10^7$，求n的质因数分解式

- $O(n)$预处理n以内质数，同时预处理每个合数x的最小质因子p_x
- 每次分解不断将n提取一个p_x
- 由于每提取一个质因子n就会减半，单次分解复杂度$O(\log n)$

质因数分解

给定正整数$n \le 10^7$，求n的质因数分解式

- $O(n)$预处理n以内质数，同时预处理每个合数x的最小质因子p_x
- 每次分解不断将n提取一个p_x
- 由于每提取一个质因子n就会减半，单次分解复杂度$O(\log n)$

质因数分解

给定正整数$n \le 10^7$，求n的质因数分解式

- $O(n)$预处理n以内质数，同时预处理每个合数x的最小质因子p_x
- 每次分解不断将n提取一个p_x
- 由于每提取一个质因子n就会减半，单次分解复杂度$O(\log n)$

最大公因数和最小公倍数

- a, b的最大公因数为最大的正整数c，$c|a$且$c|b$，记作$c = \gcd(a, b)$或$c = (a, b)$，特别地$(a, 0) = (0, a) = 0$
- a, b的最小公倍数为最小的正整数c，$a|c$且$b|c$，记作$c = \mathrm{lcm}(a, b)$或$c = [a, b]$

最大公因数和最小公倍数

- a, b的最大公因数为最大的正整数c，$c|a$且$c|b$，记作$c = \gcd(a, b)$或$c = (a, b)$，特别地$(a, 0) = (0, a) = 0$
- a, b的最小公倍数为最小的正整数c，$a|c$且$b|c$，记作$c = \mathrm{lcm}(a, b)$或$c = [a, b]$

最大公因数和最小公倍数

性质：

- $(a, b) = (a, b \pm a)$
- 设a, b的质因数分解式为$a = \prod_{i=1}^{n} p_i^{x_i}, b = \prod_{i=1}^{n} p_i^{y_i}$，则

$$(a, b) = \prod_{i=1}^{n} p_i^{\min\{x_i, y_i\}}$$

$$[a, b] = \prod_{i=1}^{n} p_i^{\max\{x_i, y_i\}}$$

- $(a, b) \cdot [a, b] = ab$

最大公因数和最小公倍数

性质：

- $(a,b)=(a,b\pm a)$
- 设a,b的质因数分解式为$a=\prod_{i=1}^{n}p_i^{x_i},b=\prod_{i=1}^{n}p_i^{y_i}$，则

$$(a,b)=\prod_{i=1}^{n}p_i^{\min\{x_i,y_i\}}$$

$$[a,b]=\prod_{i=1}^{n}p_i^{\max\{x_i,y_i\}}$$

- $(a,b)\cdot[a,b]=ab$

最大公因数和最小公倍数

性质：

- $(a,b)=(a,b\pm a)$
- 设a,b的质因数分解式为$a=\prod_{i=1}^{n}p_i^{x_i},b=\prod_{i=1}^{n}p_i^{y_i}$，则

$$(a,b)=\prod_{i=1}^{n}p_i^{\min\{x_i,y_i\}}$$

$$[a,b]=\prod_{i=1}^{n}p_i^{\max\{x_i,y_i\}}$$

- $(a,b)\cdot[a,b]=ab$

最大公因数和最小公倍数

辗转相除法求(a, b)——$O(\log ab)$

$$(a,b)=\begin{cases} a, & b=0, \\ (b, a \bmod b), & b \neq 0 \end{cases}$$

复杂度证明：

- 若$b \le a < 2b$，则$a \bmod b = a - b < \frac{a}{2}$
- 若$a \ge 2b$，则$a \bmod b < b \le a2$
- 因此每次递归a, b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

最大公因数和最小公倍数

辗转相除法求(a,b)——$O(\log ab)$

$$(a,b)=\begin{cases} a, & b=0, \\ (b, a \bmod b), & b \neq 0 \end{cases}$$

复杂度证明：

- 若$b \le a < 2b$，则$a \bmod b = a - b < \frac{a}{2}$
- 若$a \ge 2b$，则$a \bmod b < b \le a2$
- 因此每次递归a,b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

最大公因数和最小公倍数

辗转相除法求(a,b)——$O(\log ab)$

$$(a,b)=\begin{cases} a, & b=0, \\ (b, a \bmod b), & b\neq 0 \end{cases}$$

复杂度证明：

- 若$b\leq a<2b$，则$a \bmod b=a-b<\frac{a}{2}$
- 若$a\geq 2b$，则$a \bmod b<b\leq a2$
- 因此每次递归a,b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

最大公因数和最小公倍数

辗转相除法求(a, b)——$O(\log ab)$

$$(a, b) = \begin{cases} a, & b = 0, \\ (b, a \bmod b), & b \neq 0 \end{cases}$$

复杂度证明：

- 若$b \leq a < 2b$，则$a \bmod b = a - b < \frac{a}{2}$
- 若$a \geq 2b$，则$a \bmod b < b \leq a2$
- 因此每次递归a, b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

最大公因数和最小公倍数

辗转相除法求(a, b)——$O(\log ab)$

$$(a, b) = \begin{cases} a, & b = 0, \\ (b, a \bmod b), & b \neq 0 \end{cases}$$

复杂度证明：

- 若$b \leq a < 2b$，则$a \bmod b = a - b < \frac{a}{2}$
- 若$a \geq 2b$，则$a \bmod b < b \leq a2$
- 因此每次递归a, b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

最大公因数和最小公倍数

辗转相除法（欧几里得算法）求(a, b)——$O(\log ab)$

$$(a,b)=\begin{cases} a, & b=0, \\ (b, a \bmod b), & b\neq 0 \end{cases}$$

复杂度证明：

- 若$b \le a < 2b$，则$a \bmod b = a - b < \frac{a}{2}$
- 若$a \ge 2b$，则$a \bmod b < b \le a2$
- 因此每次递归a, b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

最大公因数和最小公倍数

辗转相除法（欧几里得算法）求(a,b)——$O(\log ab)$

$$(a,b)=\begin{cases} a, & b=0, \\ (b, a \bmod b), & b\neq 0 \end{cases}$$

复杂度证明：

- 若$b\leq a<2b$，则$a \bmod b=a-b<\frac{a}{2}$
- 若$a\geq 2b$，则$a \bmod b<b\leq a2$
- 因此每次递归a,b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

最大公因数和最小公倍数

辗转相除法（欧几里得算法）求(a,b)——$O(\log ab)$

$$(a,b)=\begin{cases} a, & b=0, \\ (b, a \bmod b), & b\neq 0 \end{cases}$$

复杂度证明：

- 若$b\leq a<2b$，则$a \bmod b=a-b<\frac{a}{2}$
- 若$a\geq 2b$，则$a \bmod b<b\leq a2$
- 因此每次递归a,b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

最大公因数和最小公倍数

辗转相除法（欧几里得算法）求(a,b)——$O(\log ab)$

$$(a,b)=\begin{cases} a, & b=0, \\ (b, a \bmod b), & b \neq 0 \end{cases}$$

复杂度证明：

- 若$b \le a < 2b$，则$a \bmod b = a - b < \frac{a}{2}$
- 若$a \ge 2b$，则$a \bmod b < b \le a2$
- 因此每次递归a,b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

最大公因数和最小公倍数

辗转相除法（欧几里得算法）求(a,b)——$O(\log ab)$

$$(a,b)=\begin{cases} a, & b=0, \\ (b, a \bmod b), & b \neq 0 \end{cases}$$

复杂度证明：

- 若$b \leq a < 2b$，则$a \bmod b = a - b < \frac{a}{2}$
- 若$a \geq 2b$，则$a \bmod b < b \leq a2$
- 因此每次递归a, b中较大的一个最多减半

将辗转相除法卡到最慢的是Fibonacci数

例题：NOIP2009 T2 Hankson的趣味题

给定正整数a_0, a_1, b_0, b_1，求有多少个正整数x满足：
1. x和a_0的最大公约数是a_1
2. x和b_0的最小公倍数是b_1
不超过2000组数据，$a_0, a_1, b_0, b_1 \leq 2 \times 10^9$

例题：NOIP2009 T2 Hankson的趣味题

质因数分解

每个质因数独立，考虑某个b_1包含的质因数p，设x, a_0, a_1, b_0, b_1中p的次数分别为t, u_0, u_1, v_0, v_1

$$\min\{t, u_0\} = u_1$$

$$\max\{t, v_0\} = v_1$$

求出对应t的区间，所有质因子的区间长度乘积就是答案

预处理：筛$\sqrt{2 \times 10^9}$以内质数，用$O(\frac{\sqrt{b_1}}{\log b_1})$的时间分解$b_1$，再用$b_1$的质因子分解$a_0, a_1, b_0$即可

复杂度$O(n\frac{\sqrt{b_1}}{\log b_1})$

例题：NOIP2009 T2 Hankson的趣味题

质因数分解

每个质因数独立，考虑某个b_1包含的质因数p，设x, a_0, a_1, b_0, b_1中p的次数分别为t, u_0, u_1, v_0, v_1

$$\min\{t, u_0\} = u_1$$

$$\max\{t, v_0\} = v_1$$

求出对应t的区间，所有质因子的区间长度乘积就是答案

预处理：筛$\sqrt{2 \times 10^9}$以内质数，用$O(\frac{\sqrt{b_1}}{\log b_1})$的时间分解$b_1$，再用$b_1$的质因子分解$a_0, a_1, b_0$即可

复杂度$O(n\frac{\sqrt{b_1}}{\log b_1})$

例题：NOIP2009 T2 Hankson的趣味题

质因数分解

每个质因数独立，考虑某个b_1包含的质因数p，设x, a_0, a_1, b_0, b_1中p的次数分别为t, u_0, u_1, v_0, v_1

$$\min\{t, u_0\} = u_1$$

$$\max\{t, v_0\} = v_1$$

求出对应t的区间，所有质因子的区间长度乘积就是答案

预处理：筛$\sqrt{2 \times 10^9}$以内质数，用$O(\frac{\sqrt{b_1}}{\log b_1})$的时间分解$b_1$，再用$b_1$的质因子分解$a_0, a_1, b_0$即可

复杂度$O(n\frac{\sqrt{b_1}}{\log b_1})$

例题：NOIP2009 T2 Hankson的趣味题

质因数分解

每个质因数独立，考虑某个b_1包含的质因数p，设x, a_0, a_1, b_0, b_1中p的次数分别为t, u_0, u_1, v_0, v_1

$$\min\{t, u_0\} = u_1$$

$$\max\{t, v_0\} = v_1$$

求出对应t的区间，所有质因子的区间长度乘积就是答案

预处理：筛$\sqrt{2\times 10^9}$以内质数，用$O(\frac{\sqrt{b_1}}{\log b_1})$的时间分解$b_1$，再用$b_1$的质因子分解$a_0, a_1, b_0$即可

复杂度$O(n\frac{\sqrt{b_1}}{\log b_1})$

例题：NOIP2009 T2 Hankson的趣味题

质因数分解
每个质因数独立，考虑某个b_1包含的质因数p，
设x, a_0, a_1, b_0, b_1中p的次数分别为t, u_0, u_1, v_0, v_1

$$\min\{t, u_0\} = u_1$$

$$\max\{t, v_0\} = v_1$$

求出对应t的区间，所有质因子的区间长度乘积就是答案
预处理：筛$\sqrt{2 \times 10^9}$以内质数，用$O(\frac{\sqrt{b_1}}{\log b_1})$的时间分解$b_1$，再用$b_1$的质因子分解$a_0, a_1, b_0$即可
复杂度$O(n\frac{\sqrt{b_1}}{\log b_1})$

例题：NOIP2009 T2 Hankson的趣味题

质因数分解

每个质因数独立，考虑某个b_1包含的质因数p，设x, a_0, a_1, b_0, b_1中p的次数分别为t, u_0, u_1, v_0, v_1

$$\min\{t, u_0\} = u_1$$

$$\max\{t, v_0\} = v_1$$

求出对应t的区间，所有质因子的区间长度乘积就是答案

预处理：筛$\sqrt{2 \times 10^9}$以内质数，用$O(\frac{\sqrt{b_1}}{\log b_1})$的时间分解$b_1$，再用$b_1$的质因子分解$a_0, a_1, b_0$即可

复杂度$O(n\frac{\sqrt{b_1}}{\log b_1})$

例题：orzcjk(1)

给定正整数a, b，判断a是否包含b的所有质因子，T组数据
$a, b \leq 10^{18}, T \leq 10^5$

例题：orzcjk(1)

质因数分解会TLE

用$P(a)$表示a的质因子集合，那么$P(b) \subseteq P(a) \Leftrightarrow P(b) \subseteq P(a')$，其中$a' = \gcd(a, b)$

显然$a'|b$，于是不断把b除以a'直到a'不整除b，记此时的b为b'

那么$P(b) \subseteq P(a') \Leftrightarrow P(b') \subseteq P(a')$

这样递归下去就$O(\log^2 ab)$解决了

例题：orzcjk(1)

质因数分解会TLE

用$P(a)$表示a的质因子集合，那么$P(b) \subseteq P(a) \Leftrightarrow P(b) \subseteq P(a')$，其中$a' = \gcd(a, b)$

显然$a'|b$，于是不断把b除以a'直到a'不整除b，记此时的b为b'

那么$P(b) \subseteq P(a') \Leftrightarrow P(b') \subseteq P(a')$

这样递归下去就$O(\log^2 ab)$解决了

例题：orzcjk(1)

质因数分解会TLE

用$P(a)$表示a的质因子集合，那么$P(b) \subseteq P(a) \Leftrightarrow P(b) \subseteq P(a')$，其中$a' = \gcd(a, b)$

显然$a'|b$，于是不断把b除以a'直到a'不整除b，记此时的b为b'

那么$P(b) \subseteq P(a') \Leftrightarrow P(b') \subseteq P(a')$

这样递归下去就$O(\log^2 ab)$解决了

例题：orzcjk(1)

质因数分解会TLE
用$P(a)$表示a的质因子集合，那么$P(b) \subseteq P(a) \Leftrightarrow P(b) \subseteq P(a')$，其中$a' = \gcd(a, b)$
显然$a'|b$，于是不断把b除以a'直到a'不整除b，记此时的b为b'
那么$P(b) \subseteq P(a') \Leftrightarrow P(b') \subseteq P(a')$
这样递归下去就$O(\log^2 ab)$解决了

例题：orzcjk(1)

质因数分解会TLE

用$P(a)$表示a的质因子集合，那么$P(b) \subseteq P(a) \Leftrightarrow P(b) \subseteq P(a')$，其中$a' = \gcd(a, b)$

显然$a'|b$，于是不断把b除以a'直到a'不整除b，记此时的b为b'

那么$P(b) \subseteq P(a') \Leftrightarrow P(b') \subseteq P(a')$

这样递归下去就$O(\log^2 ab)$解决了

一次不定方程

已知整数a, b, c，求关于x, y的方程$ax + by = c$的整数解

显然当(a, b)不整除c时无解

否则，可以先求出方程$\frac{a}{(a,b)}x + \frac{b}{(a,b)}y = 1$的解，再推出原方程的解

如果$(a, b) = 1$，那么当方程有特解$x = x_0, y = y_0$时，方程的通解为

$$x = x_0 + bt, y = y_0 - at, t \in \mathbf{Z}$$

一次不定方程

已知整数a, b, c，求关于x, y的方程$ax + by = c$的整数解

显然当(a, b)不整除c时无解

否则，可以先求出方程$\frac{a}{(a,b)}x + \frac{b}{(a,b)}y = 1$的解，再推出原方程的解

如果$(a, b) = 1$，那么当方程有特解$x = x_0, y = y_0$时，方程的通解为

$$x = x_0 + bt, y = y_0 - at, t \in \mathbf{Z}$$

一次不定方程

已知整数a, b, c，求关于x, y的方程$ax + by = c$的整数解

显然当(a, b)不整除c时无解

否则，可以先求出方程$\frac{a}{(a,b)}x + \frac{b}{(a,b)}y = 1$的解，再推出原方程的解

如果$(a, b) = 1$，那么当方程有特解$x = x_0, y = y_0$时，方程的通解为

$$x = x_0 + bt, y = y_0 - at, t \in \mathbf{Z}$$

扩展欧几里得算法

扩展欧几里得算法求特解

```
void exgcd(int a,int b,int&x,int&y){
    if(!b)x=1,y=0;
    else exgcd(b,a%b,y,x),y-=a/b*x;
}
```

复杂度$O(\log ab)$

同余方程组

给定方程组

$$\begin{cases} x \equiv x_1 \pmod p_1 \\ x \equiv x_2 \pmod p_2 \\ \cdots \\ x \equiv x_n \pmod p_n \end{cases}$$

求最小非负整数解

$p_1, p_2, \cdots, p_n$两两不同，$\prod p_i \leq 10^{18}$

同余方程组

每次合并两个方程

$$x \equiv x_1 \pmod p_1$$

$$x \equiv x_2 \pmod p_2$$

可得关于u, v的方程

$$p_1 u + x_1 = p_2 v + x_2$$

从而合并后得到一个新的关于x的方程$x \equiv x' \pmod{p_1 p_2}$

一共合并$n-1$次，复杂度$O(n \log \prod p_i)$

同余方程组

每次合并两个方程

$$x \equiv x_1 \pmod p_1$$

$$x \equiv x_2 \pmod p_2$$

可得关于u, v的方程

$$p_1 u + x_1 = p_2 v + x_2$$

从而合并后得到一个新的关于x的方程$x \equiv x' \pmod{p_1 p_2}$

一共合并$n-1$次，复杂度$O(n \log \prod p_i)$

乘法逆元

如果b与p互质，那么a除以b可以认为是a乘b^{-1}

其中b^{-1}是b在模p意义下的乘法逆元，即$bx \equiv 1 \pmod{p}$的解

可以用exgcd求乘法逆元，$O(\log p)$

乘法逆元

如果b与p互质，那么a除以b可以认为是a乘b^{-1}
其中b^{-1}是b在模p意义下的乘法逆元，即$bx \equiv 1 \pmod{p}$的解
可以用exgcd求乘法逆元，$O(\log p)$

乘法逆元

如果b与p互质，那么a除以b可以认为是a乘b^{-1}

其中b^{-1}是b在模p意义下的乘法逆元，即$bx \equiv 1 \pmod{p}$的解

可以用exgcd求乘法逆元，$O(\log p)$

费马小定理

对于质数p和整数a，有

$$a^p \equiv a \pmod{p}$$

对于质数p和整数a（不是p的倍数），$a^{-1} \bmod p$也可以基于费马小定理用快速幂求：$a^{-1} \bmod p = a^{p-2} \bmod p$

费马小定理

对于质数p和整数a，有

$$a^p \equiv a \pmod{p}$$

对于质数p和整数a（不是p的倍数），$a^{-1} \bmod p$也可以基于费马小定理用快速幂求：$a^{-1} \bmod p = a^{p-2} \bmod p$

例：math

给定质数p和整数$1 \le a < p$，求最小的x满足$a^x \equiv 1 \pmod p$

$1 \le a, p \le 10^9$

例：math

给定质数p和整数$1 \leq a < p$，求最小的x满足$a^x \equiv 1 \pmod{p}$
$1 \leq a, p \leq 10^9$

例：math

$a^{p-1} \equiv 1 \pmod{p}$，因此答案$x|p-1$

$O(\sqrt{p})$枚举p的因数，快速幂判断即可

复杂度$O(\sqrt{p}\log p)$

例：math

$a^{p-1} \equiv 1 \pmod p$，因此答案$x|p-1$
$O(\sqrt{p})$枚举p的因数，快速幂判断即可
复杂度$O(\sqrt{p}\log p)$

例：math

$a^{p-1} \equiv 1 \pmod{p}$，因此答案$x|p-1$
$O(\sqrt{p})$枚举p的因数，快速幂判断即可
复杂度$O(\sqrt{p}\log p)$

原根

特别地，如果在上例中，a的答案是$p-1$，即$a^0, a^1, \cdots, a^{p-2}$互不相同，则称$a$是$p$的原根

质数的原根很多，最小的原根比较小，因此对于int范围内的数据，可以通过暴力枚举$g=1,2,\cdots$然后$O(\sqrt{p}\log p)$判断来找出p的一个原根

原根

特别地，如果在上例中，a的答案是$p-1$，
即$a^0, a^1, \cdots, a^{p-2}$互不相同，则称$a$是$p$的原根

质数的原根很多，最小的原根比较小，因此对于int范围内的数据，可以通过暴力枚举$g = 1, 2, \cdots$然后$O(\sqrt{p}\log p)$判断来找出p的一个原根

例：计算器

给定质数p和整数$1 \le a, b < p$，求最小的x满足$a^x \equiv b \pmod{p}$

$1 \le a, p \le 10^9$

例：计算器

给定质数p和整数$1 \le a, b < p$，求最小的x满足$a^x \equiv b \pmod{p}$

$1 \le a, p \le 10^9$

例：计算器

双向搜索

设$t=\lceil\sqrt{p}\rceil$，将答案除以t，得到$x=tq+r$，$0\le q,r<t$

$a^{tq+r}\equiv b\Leftrightarrow a^{tq}\equiv ba^{-r}$

构造

$$A=[a^0,a^t,a^{2t},\cdots,a^{(t-1)t}]$$

$$B=[b,ba^{-1},ba^{-2},\cdots,ba^{-t+1}]$$

查找数组A,B中相同元素，用sort或者hash表实现

复杂度$O(\sqrt{p}\log p)$或$O(\sqrt{p})$

这个算法称为BSGS（大步小步）

例：计算器

双向搜索

设$t=\lceil\sqrt{p}\rceil$，将答案除以t，得到$x=tq+r$，$0\leq q,r<t$

$a^{tq+r}\equiv b\Leftrightarrow a^{tq}\equiv ba^{-r}$

构造

$$A=[a^0,a^t,a^{2t},\cdots,a^{(t-1)t}]$$

$$B=[b,ba^{-1},ba^{-2},\cdots,ba^{-t+1}]$$

查找数组A,B中相同元素，用sort或者hash表实现

复杂度$O(\sqrt{p}\log p)$或$O(\sqrt{p})$

这个算法称为BSGS（大步小步）

例：计算器

双向搜索

设$t=\lceil\sqrt{p}\rceil$，将答案除以t，得到$x=tq+r$，$0\le q,r<t$

$a^{tq+r}\equiv b\Leftrightarrow a^{tq}\equiv ba^{-r}$

构造

$$A=[a^0,a^t,a^{2t},\cdots,a^{(t-1)t}]$$

$$B=[b,ba^{-1},ba^{-2},\cdots,ba^{-t+1}]$$

查找数组A,B中相同元素，用sort或者hash表实现

复杂度$O(\sqrt{p}\log p)$或$O(\sqrt{p})$

这个算法称为BSGS（大步小步）

例：计算器

双向搜索

设$t = \lceil \sqrt{p} \rceil$，将答案除以$t$，得到$x = tq + r$，$0 \le q, r < t$

$a^{tq+r} \equiv b \Leftrightarrow a^{tq} \equiv ba^{-r}$

构造

$$A = [a^0, a^t, a^{2t}, \cdots, a^{(t-1)t}]$$

$$B = [b, ba^{-1}, ba^{-2}, \cdots, ba^{-t+1}]$$

查找数组A, B中相同元素，用sort或者hash表实现

复杂度$O(\sqrt{p}\log p)$或$O(\sqrt{p})$

这个算法称为BSGS（大步小步）

例：计算器

双向搜索

设$t=\lceil\sqrt{p}\rceil$，将答案除以t，得到$x=tq+r$，$0\le q,r<t$

$a^{tq+r}\equiv b\Leftrightarrow a^{tq}\equiv ba^{-r}$

构造

$$A=[a^0,a^t,a^{2t},\cdots,a^{(t-1)t}]$$

$$B=[b,ba^{-1},ba^{-2},\cdots,ba^{-t+1}]$$

查找数组A,B中相同元素，用sort或者hash表实现

复杂度$O(\sqrt{p}\log p)$或$O(\sqrt{p})$

这个算法称为BSGS（大步小步）

例：计算器

双向搜索

设$t=\lceil\sqrt{p}\rceil$，将答案除以t，得到$x=tq+r$，$0\le q,r<t$

$a^{tq+r}\equiv b\Leftrightarrow a^{tq}\equiv ba^{-r}$

构造

$$A=[a^0,a^t,a^{2t},\cdots,a^{(t-1)t}]$$

$$B=[b,ba^{-1},ba^{-2},\cdots,ba^{-t+1}]$$

查找数组A,B中相同元素，用sort或者hash表实现

复杂度$O(\sqrt{p}\log p)$或$O(\sqrt{p})$

这个算法称为BSGS（大步小步）

例：计算器（改）

给定质数p和整数$1 \le a, b < p$，求最小的x满足$x^a \equiv b \pmod p$

$1 \le a, p \le 10^9$

例：计算器（改）

给定质数p和整数$1 \leq a, b < p$，求最小的x满足$x^a \equiv b \pmod p$

$1 \leq a, p \leq 10^9$

例：计算器（改）

找出p的原根g

当$x \not\equiv 0 \pmod p$时，把x表示为g^x

$$g^{ax} \equiv b \pmod p$$

用g^a替代之前的a做BSGS即可

例：计算器（改）

找出p的原根g
当$x \not\equiv 0 \pmod{p}$时，把x表示为g^x

$$g^{ax} \equiv b \pmod{p}$$

用g^a替代之前的a做BSGS即可

例：计算器（改）

找出p的原根g
当$x \not\equiv 0 \pmod{p}$时，把x表示为g^x

$$g^{ax} \equiv b \pmod{p}$$

用g^a替代之前的a做BSGS即可

例：计算器（改）

找出p的原根g
当$x \not\equiv 0 \pmod p$时，把x表示为g^x

$$g^{ax} \equiv b \pmod p$$

用g^a替代之前的a做BSGS即可

数论函数

常用数论函数：

- 取整函数：$\lfloor x \rfloor$表示不大于x的最大整数
- 除数函数：$\sigma_k(x) = \sum_{d|x} d^k$
- 欧拉函数：$\varphi(x) = \sum_{i=1}^{x} [(i,x) = 1]$
- 莫比乌斯函数：$\mu(x) = \begin{cases} (-1)^k, & x = p_1 p_2 \cdots p_k \\ 0, & \text{otherwise} \end{cases}$，$p_1, \cdots, p_k$为$k$（$k \geq 0$）个互不相同的质数

数论函数

常用数论函数：

- 取整函数：$\lfloor x \rfloor$表示不大于x的最大整数
- 除数函数：$\sigma_k(x) = \sum_{d|x} d^k$
- 欧拉函数：$\varphi(x) = \sum_{i=1}^{x} [(i,x) = 1]$
- 莫比乌斯函数：$\mu(x) = \begin{cases} (-1)^k, & x = p_1 p_2 \cdots p_k \\ 0, & \text{otherwise} \end{cases}$，$p_1, \cdots, p_k$为$k$（$k \geq 0$）个互不相同的质数

数论函数

常用数论函数：

- 取整函数：$\lfloor x \rfloor$表示不大于x的最大整数
- 除数函数：$\sigma_k(x) = \sum_{d|x} d^k$
- 欧拉函数：$\varphi(x) = \sum_{i=1}^{x} [(i,x) = 1]$
- 莫比乌斯函数：$\mu(x) = \begin{cases} (-1)^k, & x = p_1 p_2 \cdots p_k \\ 0, & \text{otherwise} \end{cases}$，$p_1, \cdots, p_k$为$k$（$k \geq 0$）个互不相同的质数

数论函数

常用数论函数：

- 取整函数：$\lfloor x \rfloor$表示不大于x的最大整数
- 除数函数：$\sigma_k(x) = \sum_{d|x} d^k$
- 欧拉函数：$\varphi(x) = \sum_{i=1}^{x} [(i,x) = 1]$
- 莫比乌斯函数：$\mu(x) = \begin{cases} (-1)^k, & x = p_1 p_2 \cdots p_k \\ 0, & \text{otherwise} \end{cases}$，$p_1, \cdots, p_k$为$k$（$k \geq 0$）个互不相同的质数

数论函数

C++的a/b并不是计算$\lfloor\frac{a}{b}\rfloor$，而是向0取整，例如$\lfloor\frac{-4}{3}\rfloor=-2$，但C++中(-4)/3的结果是-1

数论函数

设n的质因数分解式为$n=\prod_{i=1}^{m} p_i^{c_i}$

- n的因数个数

$$\sigma_0(n)=\sum_{d_1=0}^{c_1}\sum_{d_2=0}^{c_2}\cdots\sum_{d_m=0}^{c_m}1=\prod_{i=1}^{m}(c_i+1)$$

- n的因数和

$$\sigma_1(n)=\sum_{d_1=0}^{c_1}p_1^{d_1}\sum_{d_2=0}^{c_2}p_2^{d_2}\cdots\sum_{d_m=0}^{c_m}p_m^{d_m}=\prod_{i=1}^{m}(1+p_i+p_i^2+\cdots+p_i^{c_i})$$

数论函数

设n的质因数分解式为$n=\prod_{i=1}^{m} p_i^{c_i}$

- n的因数个数

$$\sigma_0(n)=\sum_{d_1=0}^{c_1}\sum_{d_2=0}^{c_2}\cdots\sum_{d_m=0}^{c_m}1=\prod_{i=1}^{m}(c_i+1)$$

- n的因数和

$$\sigma_1(n)=\sum_{d_1=0}^{c_1}p_1^{d_1}\sum_{d_2=0}^{c_2}p_2^{d_2}\cdots\sum_{d_m=0}^{c_m}p_m^{d_m}=\prod_{i=1}^{m}(1+p_i+p_i^2+\cdots+p_i^{c_i})$$

数论函数

设n的质因数分解式为$n=\prod_{i=1}^{m} p_i^{c_i}$

- n的因数个数

$$\sigma_0(n)=\sum_{d_1=0}^{c_1}\sum_{d_2=0}^{c_2}\cdots\sum_{d_m=0}^{c_m}1=\prod_{i=1}^{m}(c_i+1)$$

- n的因数和

$$\sigma_1(n)=\sum_{d_1=0}^{c_1}p_1^{d_1}\sum_{d_2=0}^{c_2}p_2^{d_2}\cdots\sum_{d_m=0}^{c_m}p_m^{d_m}=\prod_{i=1}^{m}(1+p_i+p_i^2+\cdots+p_i^{c_i})$$

例题：orzcjk (2)

给定$n \leq 10^9$，求$\sum_{i=1}^{n} \sigma_0(i)$，即$1, 2, \cdots, n$的因数个数和

例题：orzcjk (2)

正着做不好做，反过来做

$$\sum_{i=1}^{n}\sigma_0(i)=\sum_{i=1}^{n}\sum_{j=1}^{n}[j|i]=\sum_{j=1}^{n}\sum_{i=1}^{n}[j|i]=\sum_{j=1}^{n}\lfloor\frac{n}{j}\rfloor$$

$\frac{n}{j}$只有$2\sqrt{n}$种取值，每种取值的j对应一段连续区间$[l,r)$

```
for(int l=1,r;l<=n;l=r){
    r=n/(n/l)+1;
    sum+=1ll*n/l*(r-l);
}
```

复杂度$O(\sqrt{n})$

例题：orzcjk (2)

正着做不好做，反过来做

$$\sum_{i=1}^{n}\sigma_0(i)=\sum_{i=1}^{n}\sum_{j=1}^{n}[j|i]=\sum_{j=1}^{n}\sum_{i=1}^{n}[j|i]=\sum_{j=1}^{n}\lfloor\frac{n}{j}\rfloor$$

$\frac{n}{j}$只有$2\sqrt{n}$种取值，每种取值的j对应一段连续区间$[l,r)$

```
for(int l=1,r;l<=n;l=r){
    r=n/(n/l)+1;
    sum+=1ll*n/l*(r-l);
}
```

复杂度$O(\sqrt{n})$

欧拉函数

n的欧拉函数$\varphi(n)$定义为$[1,n]$内与n互质的整数个数

n	1	2	3	4	5	6	7	8	9	10	$\cdots$
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4	$\cdots$

可以用容斥原理计算$\varphi(n)$

欧拉函数

n的欧拉函数$\varphi(n)$定义为$[1,n]$内与n互质的整数个数

n	1	2	3	4	5	6	7	8	9	10	$\cdots$
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4	$\cdots$

可以用容斥原理计算$\varphi(n)$

容斥原理

对于集合A, B，有

$$|A \cup B| = |A| + |B| - |A \cap B|$$

$$|A \cap B| = |A| + |B| - |A \cup B|$$

对于集合A, B, C，有

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

$$|A \cap B \cap C| = |A| + |B| + |C| - |A \cup B| - |A \cup C| - |B \cup C| + |A \cup B \cup C|$$

容斥原理

对于集合A, B，有

$$|A \cup B| = |A| + |B| - |A \cap B|$$

$$|A \cap B| = |A| + |B| - |A \cup B|$$

对于集合A, B, C，有

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

$$|A \cap B \cap C| = |A| + |B| + |C| - |A \cup B| - |A \cup C| - |B \cup C| + |A \cup B \cup C|$$

容斥原理

设全集为U，对于n个集合$A_1, A_2, \cdots, A_n$，有

$$|U - A_1 - A_2 - \cdots - A_n| = \sum_{S \subseteq \{1,2,\cdots,n\}} (-1)^{|S|} \left| \bigcap_{i \in S} A_i \right|$$

即统计同时不满足多个约束的元素个数时，用所有元素减去满足每个约束的元素，再加上满足每两个约束的元素……

容斥原理

设全集为U，对于n个集合$A_1, A_2, \cdots, A_n$，有

$$|U - A_1 - A_2 - \cdots - A_n| = \sum_{S \subseteq \{1,2,\cdots,n\}} (-1)^{|S|} \left|\bigcap_{i \in S} A_i\right|$$

即统计同时不满足多个约束的元素个数时，用所有元素减去满足每个约束的元素，再加上满足每两个约束的元素……

容斥原理

设全集为U，对于n个集合$A_1, A_2, \cdots, A_n$，有

$$|U - A_1 - A_2 - \cdots - A_n| = \sum_{S \subseteq \{1,2,\cdots,n\}} (-1)^{|S|} \left|\bigcap_{i \in S} A_i\right|$$

即统计同时不满足多个约束的元素个数时，用所有元素减去满足每个约束的元素，再加上满足每两个约束的元素……

欧拉函数

设$n=\prod_{i=1}^{m} p_i^{c_i}$，由容斥原理

$$\varphi(n)=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\sum_{x=1}^{n}\prod_{p\in S}[p|x]$$

$$=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\frac{n}{\prod_{p\in S} p}$$

$$=n\sum_{d_1=0}^{1}(-\frac{1}{p_1})^{d_1}\sum_{d_2=0}^{1}(-\frac{1}{p_2})^{d_2}\cdots\sum_{d_m=0}^{1}(-\frac{1}{p_m})^{d_m}1=n\prod_{i=1}^{m}(1-\frac{1}{p_i})$$

可以质因数分解计算，也可以在线性筛质数时预处理$\varphi(1)\cdots\varphi(n)$

欧拉函数

设$n=\prod_{i=1}^m p_i^{c_i}$，由容斥原理

$$\varphi(n)=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\sum_{x=1}^{n}\prod_{p\in S}[p|x]$$

$$=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\frac{n}{\prod_{p\in S}p}$$

$$=n\sum_{d_1=0}^{1}(-\frac{1}{p_1})^{d_1}\sum_{d_2=0}^{1}(-\frac{1}{p_2})^{d_2}\cdots\sum_{d_m=0}^{1}(-\frac{1}{p_m})^{d_m}1=n\prod_{i=1}^{m}(1-\frac{1}{p_i})$$

可以质因数分解计算，也可以在线性筛质数时预处理$\varphi(1)\cdots\varphi(n)$

欧拉函数

设$n=\prod_{i=1}^{m}p_i^{c_i}$，由容斥原理

$$\varphi(n)=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\sum_{x=1}^{n}\prod_{p\in S}[p|x]$$

$$=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\frac{n}{\prod_{p\in S}p}$$

$$=n\sum_{d_1=0}^{1}(-\frac{1}{p_1})^{d_1}\sum_{d_2=0}^{1}(-\frac{1}{p_2})^{d_2}\cdots\sum_{d_m=0}^{1}(-\frac{1}{p_m})^{d_m}1=n\prod_{i=1}^{m}(1-\frac{1}{p_i})$$

可以质因数分解计算，也可以在线性筛质数时预处理$\varphi(1)\cdots\varphi(n)$

欧拉函数

设$n=\prod_{i=1}^{m} p_i^{c_i}$，由容斥原理

$$\varphi(n)=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\sum_{x=1}^{n}\prod_{p\in S}[p|x]$$

$$=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\frac{n}{\prod_{p\in S}p}$$

$$=n\sum_{d_1=0}^{1}(-\frac{1}{p_1})^{d_1}\sum_{d_2=0}^{1}(-\frac{1}{p_2})^{d_2}\cdots\sum_{d_m=0}^{1}(-\frac{1}{p_m})^{d_m}1=n\prod_{i=1}^{m}(1-\frac{1}{p_i})$$

可以质因数分解计算，也可以在线性筛质数时预处理$\varphi(1)\cdots\varphi(n)$

欧拉函数

设$n=\prod_{i=1}^{m}p_i^{c_i}$，由容斥原理

$$\varphi(n)=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\sum_{x=1}^{n}\prod_{p\in S}[p|x]$$

$$=\sum_{S\subseteq\{p_1,p_2,\cdots,p_m\}}(-1)^{|S|}\frac{n}{\prod_{p\in S}p}$$

$$=n\sum_{d_1=0}^{1}(-\frac{1}{p_1})^{d_1}\sum_{d_2=0}^{1}(-\frac{1}{p_2})^{d_2}\cdots\sum_{d_m=0}^{1}(-\frac{1}{p_m})^{d_m}1=n\prod_{i=1}^{m}(1-\frac{1}{p_i})$$

可以质因数分解计算，也可以在线性筛质数时预处理$\varphi(1)\cdots\varphi(n)$

欧拉函数

欧拉函数的性质：

- 对正整数n和质数p，

$$\varphi(pn)=\begin{cases}p\varphi(n), & p|n,\\ (p-1)\varphi(n), & \text{otherwise}\end{cases}$$

- 欧拉函数是积性函数，即若$(a,b)=1$，则$\varphi(ab)=\varphi(a)\varphi(b)$
- $\sum_{d|n}\varphi(d)=n$

欧拉函数

欧拉函数的性质：

- 对正整数n和质数p，

$$\varphi(pn)=\begin{cases}p\varphi(n), & p|n,\\ (p-1)\varphi(n), & \text{otherwise}\end{cases}$$

- 欧拉函数是积性函数，即若$(a,b)=1$，则$\varphi(ab)=\varphi(a)\varphi(b)$
- $\sum_{d|n}\varphi(d)=n$

欧拉函数

欧拉函数的性质：

- 对正整数n和质数p，

$$\varphi(pn) = \begin{cases} p\varphi(n), & p|n, \\ (p-1)\varphi(n), & \text{otherwise} \end{cases}$$

- 欧拉函数是积性函数，即若$(a,b)=1$，则$\varphi(ab)=\varphi(a)\varphi(b)$
- $\sum_{d|n} \varphi(d) = n$

莫比乌斯函数

莫比乌斯函数$\mu(n)$：

- n有平方因子：$\mu(n)=0$
- n无平方因子，设n有k个质因数：$\mu(n)=(-1)^k$

n	1	2	3	4	5	6	7	8	9	10	$\cdots$
$\mu(n)$	1	-1	-1	0	-1	1	-1	0	0	1	$\cdots$

性质：

- 莫比乌斯函数是积性函数，即若$(a,b)=1$，则$\mu(ab)=\mu(a)\mu(b)$
- $\sum_{d|n}\mu(d)=[n=1]$

莫比乌斯函数

莫比乌斯函数$\mu(n)$：

- n有平方因子：$\mu(n)=0$
- n无平方因子，设n有k个质因数：$\mu(n)=(-1)^k$

n	1	2	3	4	5	6	7	8	9	10	$\cdots$
$\mu(n)$	1	-1	-1	0	-1	1	-1	0	0	1	$\cdots$

性质：

- 莫比乌斯函数是积性函数，即若$(a,b)=1$，则$\mu(ab)=\mu(a)\mu(b)$
- $\sum_{d|n}\mu(d)=[n=1]$

莫比乌斯函数

莫比乌斯函数$\mu(n)$：

- n有平方因子：$\mu(n) = 0$
- n无平方因子，设n有k个质因数：$\mu(n) = (-1)^k$

n	1	2	3	4	5	6	7	8	9	10	$\cdots$
$\mu(n)$	1	-1	-1	0	-1	1	-1	0	0	1	$\cdots$

性质：

- 莫比乌斯函数是积性函数，即若$(a, b) = 1$，则$\mu(ab) = \mu(a)\mu(b)$
- $\sum_{d|n} \mu(d) = [n = 1]$

莫比乌斯函数

莫比乌斯函数$\mu(n)$：

- n有平方因子：$\mu(n)=0$
- n无平方因子，设n有k个质因数：$\mu(n)=(-1)^k$

n	1	2	3	4	5	6	7	8	9	10	$\cdots$
$\mu(n)$	1	-1	-1	0	-1	1	-1	0	0	1	$\cdots$

性质：

- 莫比乌斯函数是积性函数，即若$(a,b)=1$，则$\mu(ab)=\mu(a)\mu(b)$
- $\sum_{d|n}\mu(d)=[n=1]$

莫比乌斯函数

莫比乌斯函数$\mu(n)$：

- n有平方因子： $\mu(n) = 0$
- n无平方因子，设n有k个质因数： $\mu(n) = (-1)^k$

n	1	2	3	4	5	6	7	8	9	10	$\cdots$
$\mu(n)$	1	-1	-1	0	-1	1	-1	0	0	1	$\cdots$

性质：

- 莫比乌斯函数是积性函数，即若$(a, b) = 1$，则$\mu(ab) = \mu(a)\mu(b)$
- $\sum_{d|n} \mu(d) = [n = 1]$

莫比乌斯函数

莫比乌斯函数$\mu(n)$：

- n有平方因子：$\mu(n) = 0$
- n无平方因子，设n有k个质因数：$\mu(n) = (-1)^k$

n	1	2	3	4	5	6	7	8	9	10	$\cdots$
$\mu(n)$	1	-1	-1	0	-1	1	-1	0	0	1	$\cdots$

性质：

- 莫比乌斯函数是积性函数，即若$(a, b) = 1$，则$\mu(ab) = \mu(a)\mu(b)$
- $\sum_{d|n} \mu(d) = [n = 1]$

例：51nod 1610 路径计数

给定一个n个点m条边的带权有向无环图，有T次修改操作，每次修改一条边的边权，每次修改后输出有向无环图上路径上所有边权的gcd为1的路径数量模10^9+7

$n \le 100$，$m \le 50000$，边权≤ 100，$T \le 100$

暴力DP？$f(i,j)$表示从i出发，gcd为j的路径条数，时间$T \times (n+m) \times 100$，TLE

例：51nod 1610 路径计数

给定一个n个点m条边的带权有向无环图，有T次修改操作，每次修改一条边的边权，每次修改后输出有向无环图上路径上所有边权的gcd为1的路径数量模10^9+7

$n \le 100$，$m \le 50000$，边权≤ 100，$T \le 100$

暴力DP？$f(i,j)$表示从i出发，gcd为j的路径条数，时间$T \times (n+m) \times 100$，TLE

例：51nod 1610 路径计数

给定一个n个点m条边的带权有向无环图，有T次修改操作，每次修改一条边的边权，每次修改后输出有向无环图上路径上所有边权的gcd为1的路径数量模10^9+7

$n \leq 100$，$m \leq 50000$，边权≤ 100，$T \leq 100$

暴力DP？$f(i,j)$表示从i出发，gcd为j的路径条数，时间$T \times (n+m) \times 100$，TLE

例：51nod 1610 路径计数

根据$\sum_{d|n}\mu(d)=[n=1]$，可得

$$\sum_{path}[\gcd(path)=1]=\sum_{path}\sum_{d|e,\forall e\in path}\mu(d)=\sum_{d}\mu(d)\sum_{path\in G_d}1$$

即枚举$d=1,2,\cdots,100$，统计图G_d中的路径条数，乘上$\mu(d)$贡献答案

其中G_d为取出所有边权为d的倍数的边构成的图

因为100以内的数最多只有3个质因数，所以每条边最多属于$2^3=8$个G_d（$\mu(d)\neq 0$），总边数不超过$8m$

时间效率$T\times(100n+8m)=4.1\times 10^7$，可以通过

例：51nod 1610 路径计数

根据$\sum_{d|n}\mu(d)=[n=1]$，可得

$$\sum_{path}[\gcd(path)=1]=\sum_{path}\sum_{d|e,\forall e\in path}\mu(d)=\sum_{d}\mu(d)\sum_{path\in G_d}1$$

即枚举$d=1,2,\cdots,100$，统计图G_d中的路径条数，乘上$\mu(d)$贡献答案

其中G_d为取出所有边权为d的倍数的边构成的图

因为100以内的数最多只有3个质因数，所以每条边最多属于$2^3=8$个G_d（$\mu(d)\neq 0$），总边数不超过$8m$

时间效率$T\times(100n+8m)=4.1\times 10^7$，可以通过

例：51nod 1610 路径计数

根据$\sum_{d|n}\mu(d)=[n=1]$，可得

$$\sum_{path}[\gcd(path)=1]=\sum_{path}\sum_{d|e,\forall e\in path}\mu(d)=\sum_{d}\mu(d)\sum_{path\in G_d}1$$

即枚举$d=1,2,\cdots,100$，统计图G_d中的路径条数，乘上$\mu(d)$贡献答案

其中G_d为取出所有边权为d的倍数的边构成的图

因为100以内的数最多只有3个质因数，所以每条边最多属于$2^3=8$个G_d（$\mu(d)\neq 0$），总边数不超过$8m$

时间效率$T\times(100n+8m)=4.1\times 10^7$，可以通过

莫比乌斯反演

μ函数用于莫比乌斯反演的应用十分广泛

莫比乌斯反演本质是一种容斥

我们知道，如果数论函数$f(n)$满足$\sum_{d|n} f(d) = [n = 1]$，那么$f(n) = \mu(n)$

如果数论函数$f(n)$满足$\sum_{d|n} f(d) = g(n)$，那么

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

莫比乌斯反演

- μ函数用于莫比乌斯反演的应用十分广泛
- 莫比乌斯反演本质是一种容斥
- 我们知道，如果数论函数$f(n)$满足$\sum_{d|n} f(d) = [n = 1]$，那么$f(n) = \mu(n)$
- 如果数论函数$f(n)$满足$\sum_{d|n} f(d) = g(n)$，那么

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

莫比乌斯反演

μ函数用于莫比乌斯反演的应用十分广泛

莫比乌斯反演本质是一种容斥

我们知道，如果数论函数$f(n)$满足$\sum_{d|n} f(d) = [n = 1]$，那么$f(n) = \mu(n)$

如果数论函数$f(n)$满足$\sum_{d|n} f(d) = g(n)$，那么

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

莫比乌斯反演

μ函数用于莫比乌斯反演的应用十分广泛

莫比乌斯反演本质是一种容斥

我们知道，如果数论函数$f(n)$满足$\sum_{d|n} f(d) = [n = 1]$，那么$f(n) = \mu(n)$

如果数论函数$f(n)$满足$\sum_{d|n} f(d) = g(n)$，那么

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

莫比乌斯反演

μ函数用于莫比乌斯反演的应用十分广泛

莫比乌斯反演本质是一种容斥

我们知道，如果数论函数$f(n)$满足$\sum_{d|n} f(d) = [n = 1]$，那么$f(n) = \mu(n)$

如果数论函数$f(n)$满足$\sum_{d|n} f(d) = g(n)$，那么

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

例：LCM

给定正整数$n, m \leq 10^7$，求$\sum_{i=1}^{n}\sum_{j=1}^{m}\mathrm{lcm}(i,j)$

T组询问（$T \leq 1000$）

例：LCM

$$\sum_{i=1}^{n}\sum_{j=1}^{m}[i,j]=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{ij}{(i,j)}=\sum_{i=1}^{n}i\sum_{j=1}^{m}j\sum_{d|(i,j)}f(d)$$

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

其中f函数满足$\sum_{d|n}f(d)=\frac{1}{n}$

例：LCM

设质因数分解式$n=\prod_{i=1}^{t}p_i^{c_i}$，则

$$f(n)=\frac{1}{n}\sum_{d_1=0}^{1}(-p_1)^{d_1}\sum_{d_2=0}^{1}(-p_2)^{d_2}\cdots\sum_{d_t=0}^{1}(-p_t)^{d_t}=\frac{1}{n}\prod_{i=1}^{t}(1-p_i)$$

线性筛预处理$nf(n)$，$n=1,2,\cdots,10^7$

求值时，$\lfloor\frac{n}{d}\rfloor,\lfloor\frac{m}{d}\rfloor$只有$\sqrt{n}+\sqrt{m}+\sqrt{\min\{n,m\}}$段

复杂度$O(n+T(\sqrt{n}+\sqrt{m}))$

例：LCM

设质因数分解式$n=\prod_{i=1}^{t}p_i^{c_i}$，则

$$f(n)=\frac{1}{n}\sum_{d_1=0}^{1}(-p_1)^{d_1}\sum_{d_2=0}^{1}(-p_2)^{d_2}\cdots\sum_{d_t=0}^{1}(-p_t)^{d_t}=\frac{1}{n}\prod_{i=1}^{t}(1-p_i)$$

线性筛预处理$nf(n)$，$n=1,2,\cdots,10^7$

求值时，$\lfloor\frac{n}{d}\rfloor,\lfloor\frac{m}{d}\rfloor$只有$\sqrt{n}+\sqrt{m}+\sqrt{\min\{n,m\}}$段

复杂度$O(n+T(\sqrt{n}+\sqrt{m}))$

例：LCM

设质因数分解式$n=\prod_{i=1}^{t} p_i^{c_i}$，则

$$f(n)=\frac{1}{n}\sum_{d_1=0}^{1}(-p_1)^{d_1}\sum_{d_2=0}^{1}(-p_2)^{d_2}\cdots\sum_{d_t=0}^{1}(-p_t)^{d_t}=\frac{1}{n}\prod_{i=1}^{t}(1-p_i)$$

线性筛预处理$nf(n)$，$n=1,2,\cdots,10^7$
求值时，$\lfloor\frac{n}{d}\rfloor,\lfloor\frac{m}{d}\rfloor$只有$\sqrt{n}+\sqrt{m}+\sqrt{\min\{n,m\}}$段
复杂度$O(n+T(\sqrt{n}+\sqrt{m}))$

例：最简分数

给定$n \leq 10^9$，求$(0,1]$内有多少个不同的最简分数$\frac{p}{q}$，满足$1 \leq q \leq n$

等价于$\sum_{q=1}^{n}\sum_{p=1}^{q}[(p,q)=1]=\sum_{i=1}^{n}\varphi(i)$

线性筛预处理$\varphi(1)\cdots\varphi(n)$，复杂度$O(n)$

更优的做法？

例：最简分数

给定$n \le 10^9$，求$(0,1]$内有多少个不同的最简分数$\frac{p}{q}$，满足$1 \le q \le n$

等价于$\sum_{q=1}^{n}\sum_{p=1}^{q}[(p,q)=1] = \sum_{i=1}^{n}\varphi(i)$

线性筛预处理$\varphi(1)\cdots\varphi(n)$，复杂度$O(n)$

更优的做法?

例：最简分数

给定$n \le 10^9$，求$(0,1]$内有多少个不同的最简分数$\frac{p}{q}$，满足$1 \le q \le n$

等价于$\sum_{q=1}^{n}\sum_{p=1}^{q}[(p,q)=1]=\sum_{i=1}^{n}\varphi(i)$

线性筛预处理$\varphi(1)\cdots\varphi(n)$，复杂度$O(n)$

更优的做法？

例：最简分数

用$S(n)$表示答案

$$S(n)=\sum_{q=1}^{n}\sum_{p=1}^{q}1-\sum_{d=2}^{n}\sum_{q=1}^{n}\sum_{p=1}^{q}[(p,q)=d]$$

$$=\frac{n(n+1)}{2}-\sum_{d=2}^{n}\sum_{q=1}^{\lfloor\frac{n}{d}\rfloor}\sum_{p=1}^{q}[(p,q)=1]=\frac{n(n+1)}{2}-\sum_{d=2}^{n}S(\lfloor\frac{n}{d}\rfloor)$$

$\lfloor\frac{n}{d}\rfloor$只有$O(\sqrt{n})$种取值，记忆化搜索的复杂度为

$$O(\sqrt{n}+\sqrt{\frac{n}{2}}+\sqrt{\frac{n}{3}}+\cdots+\sqrt{\sqrt{n}}+\sqrt{\sqrt{n-1}}+\cdots+1)=O(n^{\frac{3}{4}})$$

例：最简分数

用$S(n)$表示答案

$$S(n)=\sum_{q=1}^{n}\sum_{p=1}^{q}1-\sum_{d=2}^{n}\sum_{q=1}^{n}\sum_{p=1}^{q}[(p,q)=d]$$

$$=\frac{n(n+1)}{2}-\sum_{d=2}^{n}\sum_{q=1}^{\lfloor\frac{n}{d}\rfloor}\sum_{p=1}^{q}[(p,q)=1]=\frac{n(n+1)}{2}-\sum_{d=2}^{n}S(\lfloor\frac{n}{d}\rfloor)$$

$\lfloor\frac{n}{d}\rfloor$只有$O(\sqrt{n})$种取值，记忆化搜索的复杂度为

$$O(\sqrt{n}+\sqrt{\frac{n}{2}}+\sqrt{\frac{n}{3}}+\cdots+\sqrt{\sqrt{n}}+\sqrt{\sqrt{n}-1}+\cdots+1)=O(n^{\frac{3}{4}})$$

例：最简分数

用$S(n)$表示答案

$$S(n)=\sum_{q=1}^{n}\sum_{p=1}^{q}1-\sum_{d=2}^{n}\sum_{q=1}^{n}\sum_{p=1}^{q}[(p,q)=d]$$

$$=\frac{n(n+1)}{2}-\sum_{d=2}^{n}\sum_{q=1}^{\lfloor\frac{n}{d}\rfloor}\sum_{p=1}^{q}[(p,q)=1]=\frac{n(n+1)}{2}-\sum_{d=2}^{n}S(\lfloor\frac{n}{d}\rfloor)$$

$\lfloor\frac{n}{d}\rfloor$只有$O(\sqrt{n})$种取值，记忆化搜索的复杂度为

$$O(\sqrt{n}+\sqrt{\frac{n}{2}}+\sqrt{\frac{n}{3}}+\cdots+\sqrt{\sqrt{n}}+\sqrt{\sqrt{n}-1}+\cdots+1)=O(n^{\frac{3}{4}})$$

例：最简分数

用$S(n)$表示答案

$$S(n) = \sum_{q=1}^{n}\sum_{p=1}^{q} 1 - \sum_{d=2}^{n}\sum_{q=1}^{n}\sum_{p=1}^{q}[(p,q)=d]$$

$$= \frac{n(n+1)}{2} - \sum_{d=2}^{n}\sum_{q=1}^{\lfloor\frac{n}{d}\rfloor}\sum_{p=1}^{q}[(p,q)=1] = \frac{n(n+1)}{2} - \sum_{d=2}^{n} S(\lfloor\frac{n}{d}\rfloor)$$

$\lfloor\frac{n}{d}\rfloor$只有$O(\sqrt{n})$种取值，记忆化搜索的复杂度为

$$O(\sqrt{n} + \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{3}} + \cdots + \sqrt{\sqrt{n}} + \sqrt{\sqrt{n}-1} + \cdots + 1) = O(n^{\frac{3}{4}})$$

例：最简分数

用$S(n)$表示答案

$$S(n)=\sum_{q=1}^{n}\sum_{p=1}^{q}1-\sum_{d=2}^{n}\sum_{q=1}^{n}\sum_{p=1}^{q}[(p,q)=d]$$

$$=\frac{n(n+1)}{2}-\sum_{d=2}^{n}\sum_{q=1}^{\lfloor\frac{n}{d}\rfloor}\sum_{p=1}^{q}[(p,q)=1]=\frac{n(n+1)}{2}-\sum_{d=2}^{n}S(\lfloor\frac{n}{d}\rfloor)$$

$\lfloor\frac{n}{d}\rfloor$只有$O(\sqrt{n})$种取值，记忆化搜索的复杂度为

$$O(\sqrt{n}+\sqrt{\frac{n}{2}}+\sqrt{\frac{n}{3}}+\cdots+\sqrt{\sqrt{n}}+\sqrt{\sqrt{n}-1}+\cdots+1)=O(n^{\frac{3}{4}})$$

例：最简分数

进一步优化：

- 设一个上界B，用线性筛处理出$S(1), S(2), \cdots, S(B)$，剩下部分记忆化搜索
- 当$B = n^{\frac{2}{3}}$时复杂度达到最优$O(n^{\frac{2}{3}})$

杜教筛

可以换一种角度理解刚才的问题

根据$\sum_{d|i}\varphi(d)=i$，有

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

又因为

$$\sum_{i=1}^{n}\sum_{d|i}\varphi(d)=\sum_{d=1}^{n}\sum_{i=1}^{n}[id\le n]\varphi(d)=\sum_{i=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{i}\rfloor}\varphi(d)=\sum_{i=1}^{n}S(\lfloor\frac{n}{i}\rfloor)$$

所以

$$S(n)=\frac{n(n+1)}{2}-\sum^{n}S(\lfloor\frac{n}{i}\rfloor)$$

杜教筛

可以换一种角度理解刚才的问题

根据$\sum_{d|i}\varphi(d)=i$，有

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

又因为

$$\sum_{i=1}^{n}\sum_{d|i}\varphi(d)=\sum_{d=1}^{n}\sum_{i=1}^{n}[id\le n]\varphi(d)=\sum_{i=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{i}\rfloor}\varphi(d)=\sum_{i=1}^{n}S(\lfloor\frac{n}{i}\rfloor)$$

所以

$$S(n)=\frac{n(n+1)}{2}-\sum^{n}S(\lfloor\frac{n}{i}\rfloor)$$

杜教筛

可以换一种角度理解刚才的问题

根据$\sum_{d|i}\varphi(d)=i$，有

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

又因为

$$\sum_{i=1}^{n}\sum_{d|i}\varphi(d)=\sum_{d=1}^{n}\sum_{i=1}^{n}[id\le n]\varphi(d)=\sum_{i=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{i}\rfloor}\varphi(d)=\sum_{i=1}^{n}S(\lfloor\frac{n}{i}\rfloor)$$

所以

$$S(n)=\frac{n(n+1)}{2}-\sum^{n}S(\lfloor\frac{n}{i}\rfloor)$$

杜教筛

可以换一种角度理解刚才的问题

根据$\sum_{d|i}\varphi(d)=i$，有

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

又因为

$$\sum_{i=1}^{n}\sum_{d|i}\varphi(d)=\sum_{d=1}^{n}\sum_{i=1}^{n}[id\leq n]\varphi(d)=\sum_{i=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{i}\rfloor}\varphi(d)=\sum_{i=1}^{n}S(\lfloor\frac{n}{i}\rfloor)$$

所以

$$S(n)=\frac{n(n+1)}{2}-\sum^{n}S(\lfloor\frac{n}{i}\rfloor)$$

排列数与组合数

- 排列数A_n^m：将n个物品选出m个排成一行的方案数

$$A_n^m = n(n-1)(n-2)\cdots(n-m+1)$$

- 组合数C_n^m：将n个物品选出m个的方案数

$$C_n^m = \frac{n(n-1)(n-2)\cdots(n-m+1)}{m!}$$

组合数还可以写成二项式系数$\binom{n}{m} = C_n^m$

排列数与组合数

- 排列数A_n^m：将n个物品选出m个排成一行的方案数

$$A_n^m = n(n-1)(n-2)\cdots(n-m+1)$$

- 组合数C_n^m：将n个物品选出m个的方案数

$$C_n^m = \frac{n(n-1)(n-2)\cdots(n-m+1)}{m!}$$

组合数还可以写成二项式系数$\binom{n}{m} = C_n^m$

排列数与组合数

- 排列数A_n^m：将n个物品选出m个排成一行的方案数

$$A_n^m = n(n-1)(n-2)\cdots(n-m+1)$$

- 组合数C_n^m：将n个物品选出m个的方案数

$$C_n^m = \frac{n(n-1)(n-2)\cdots(n-m+1)}{m!}$$

组合数还可以写成二项式系数$\binom{n}{m} = C_n^m$

排列数与组合数

常用性质：

- $\binom{n}{m} = \frac{A_n^m}{m!}$
- $\binom{n}{m} = \frac{n!}{m!(n-m)!}$
- $\binom{n}{m} = \binom{n}{n-m}$
- $\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$

排列数与组合数

常用性质：

- $\binom{n}{m} = \frac{A_n^m}{m!}$
- $\binom{n}{m} = \frac{n!}{m!(n-m)!}$
- $\binom{n}{m} = \binom{n}{n-m}$
- $\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$

排列数与组合数

常用性质：

- $\binom{n}{m}=\frac{A_n^m}{m!}$
- $\binom{n}{m}=\frac{n!}{m!(n-m)!}$
- $\binom{n}{m}=\binom{n}{n-m}$
- $\binom{n}{m}=\binom{n-1}{m-1}+\binom{n-1}{m}$

排列数与组合数

常用性质：

- $\binom{n}{m} = \frac{A_n^m}{m!}$
- $\binom{n}{m} = \frac{n!}{m!(n-m)!}$
- $\binom{n}{m} = \binom{n}{n-m}$
- $\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$

排列数与组合数

常用性质：

- $\binom{n}{m} = \frac{A_n^m}{m!}$
- $\binom{n}{m} = \frac{n!}{m!(n-m)!}$
- $\binom{n}{m} = \binom{n}{n-m}$
- $\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$

例：NOIP2016 D2T1 组合数问题

给定n, m 和k，对于所有的$0 \le i \le n, 0 \le j \le \min(i, m)$ 有多少对(i, j) 满足C_i^j 是k 的倍数，t 组询问

$n, m \le 2000$，$k \le 21$，$t \le 10^4$

例：NOIP2016 D2T1 组合数问题

预处理出所有C_i^j是否为k的倍数，做二维前缀和即可单次$O(1)$查询

- 做法一：根据$\binom{n}{m} = \frac{n!}{m!(n-m)!}$，将$t$质因数分解，前缀和维护所有$i!$中每个$t$包含的质因子的次数
- 做法二：直接根据$\binom{n}{m} \bmod k = (\binom{n-1}{m-1} \bmod k + \binom{n-1}{m} \bmod k) \bmod k$递推

后者更加方便，考场上可以5分钟搞定

例：NOIP2016 D2T1 组合数问题

预处理出所有C_i^j是否为k的倍数，做二维前缀和即可单次$O(1)$查询

- 做法一：根据$\binom{n}{m} = \frac{n!}{m!(n-m)!}$，将$t$质因数分解，前缀和维护所有$i!$中每个$t$包含的质因子的次数
- 做法二：直接根据$\binom{n}{m} \bmod k = (\binom{n-1}{m-1} \bmod k + \binom{n-1}{m} \bmod k) \bmod k$递推

后者更加方便，考场上可以5分钟搞定

例：NOIP2016 D2T1 组合数问题

预处理出所有C_i^j是否为k的倍数，做二维前缀和即可单次$O(1)$查询

- 做法一：根据$\binom{n}{m}=\frac{n!}{m!(n-m)!}$，将$t$质因数分解，前缀和维护所有$i!$中每个$t$包含的质因子的次数
- 做法二：直接根据$\binom{n}{m} \bmod k = \left(\binom{n-1}{m-1} \bmod k + \binom{n-1}{m} \bmod k\right) \bmod k$递推

后者更加方便，考场上可以5分钟搞定

例：NOIP2016 D2T1 组合数问题

预处理出所有C_i^j是否为k的倍数，做二维前缀和即可单次$O(1)$查询

- 做法一：根据$\binom{n}{m}=\frac{n!}{m!(n-m)!}$，将$t$质因数分解，前缀和维护所有$i!$中每个$t$包含的质因子的次数
- 做法二：直接根据$\binom{n}{m} \bmod k=(\binom{n-1}{m-1} \bmod k+\binom{n-1}{m} \bmod k) \bmod k$递推

后者更加方便，考场上可以5分钟搞定

计算组合数

考虑以下问题：

- 对于整数$0 \le n, m \le 10^7$和质数$p \le 10^9$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \le n, m \le 10^7$和合数$p \le 10^9$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \le n, m \le 10^{18}$和质数$p \le 10^6$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \le n, m \le 10^{18}$和合数$p = \prod p_i^{c_i}$（$\max\{p_i^{c_i}\} \le 10^6$，求$\binom{n}{m} \bmod p$

计算组合数

考虑以下问题：

- 对于整数$0 \le n, m \le 10^7$和质数$p \le 10^9$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \le n, m \le 10^7$和合数$p \le 10^9$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \le n, m \le 10^{18}$和质数$p \le 10^6$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \le n, m \le 10^{18}$和合数$p = \prod p_i^{c_i}$（$\max\{p_i^{c_i}\} \le 10^6$，求$\binom{n}{m} \bmod p$

计算组合数

考虑以下问题：

- 对于整数$0 \leq n, m \leq 10^7$和质数$p \leq 10^9$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \leq n, m \leq 10^7$和合数$p \leq 10^9$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \leq n, m \leq 10^{18}$和质数$p \leq 10^6$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \leq n, m \leq 10^{18}$和合数$p = \prod p_i^{c_i}$（$\max\{p_i^{c_i}\} \leq 10^6$，求$\binom{n}{m} \bmod p$

计算组合数

考虑以下问题：

- 对于整数$0 \le n, m \le 10^7$和质数$p \le 10^9$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \le n, m \le 10^7$和合数$p \le 10^9$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \le n, m \le 10^{18}$和质数$p \le 10^6$，求$\binom{n}{m} \bmod p$
- 对于整数$0 \le n, m \le 10^{18}$和合数$p = \prod p_i^{c_i}$（$\max\{p_i^{c_i}\} \le 10^6$，求$\binom{n}{m} \bmod p$

矩阵

矩阵：按照矩形排列的表格

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \cdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}$$

设$A = (a_{i,j})$，$B = (b_{i,j})$，$C = (c_{i,j})$

- 矩阵加法$C = A + B$：$c_{i,j} = a_{i,j} + b_{i,j}$（$A, B$均为$n \times m$矩阵）
- 矩阵乘法$C = AB$：$c_{i,j} = \sum_k a_{i,k} b_{k,j}$（$A$的宽度等于$B$的高度）

矩阵

矩阵：按照矩形排列的表格

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \cdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}$$

设$A = (a_{i,j})$，$B = (b_{i,j})$，$C = (c_{i,j})$

- 矩阵加法$C = A + B$：$c_{i,j} = a_{i,j} + b_{i,j}$（$A, B$均为$n \times m$矩阵）
- 矩阵乘法$C = AB$：$c_{i,j} = \sum_k a_{i,k} b_{k,j}$（$A$的宽度等于$B$的高度）

矩阵

矩阵：按照矩形排列的表格

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \cdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}$$

设$A = (a_{i,j})$，$B = (b_{i,j})$，$C = (c_{i,j})$

- 矩阵加法$C = A + B$：$c_{i,j} = a_{i,j} + b_{i,j}$（$A, B$均为$n \times m$矩阵）
- 矩阵乘法$C = AB$：$c_{i,j} = \sum_k a_{i,k} b_{k,j}$（$A$的宽度等于$B$的高度）

矩阵

矩阵：按照矩形排列的表格

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \cdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}$$

设$A = (a_{i,j})$，$B = (b_{i,j})$，$C = (c_{i,j})$

- 矩阵加法$C = A + B$：$c_{i,j} = a_{i,j} + b_{i,j}$（$A, B$均为$n \times m$矩阵）
- 矩阵乘法$C = AB$：$c_{i,j} = \sum_k a_{i,k} b_{k,j}$（$A$的宽度等于$B$的高度）

线性方程组

线性方程组可以写成矩阵形式

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \cdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{pmatrix}$$

线性方程组

通过高斯消元可以解线性方程组

对于$i = 1, 2, \cdots, n$:

- 找到第i列的一个非0元素$a_{j,i}$（为避免精度误差，最好选绝对值最大的）
- 把第j行交换到第i行
- 对于$j = i + 1 \cdots n$，令$\forall k, a_{j,k} \leftarrow a_{j,k} - a_{i,k} \cdot \frac{a_{j,i}}{a_{i,i}}$，$b_j \leftarrow b_j - b_i \cdot \frac{a_{j,i}}{a_{i,i}}$
- 得到上三角矩阵，回代得到解(x_i)

注意也有可能找不到非0元素，这样消出的矩阵有一些空行，如果某个空行的b不为0则无解

否则如果$n = m$则有唯一解

复杂度$O(n^2 m)$

线性方程组

通过高斯消元可以解线性方程组

对于$i = 1, 2, \cdots, n$:

- 找到第i列的一个非0元素$a_{j,i}$（为避免精度误差，最好选绝对值最大的）
- 把第j行交换到第i行
- 对于$j = i + 1 \cdots n$，
  令$\forall k, a_{j,k} \leftarrow a_{j,k} - a_{i,k} \cdot \frac{a_{j,i}}{ai,i}$，$b_j \leftarrow b_j - b_i \cdot \frac{a_{j,i}}{ai,i}$
- 得到上三角矩阵，回代得到解(x_i)

注意也有可能找不到非0元素，这样消出的矩阵有一些空行，

如果某个空行的b不为0则无解

否则如果$n = m$则有唯一解

复杂度$O(n^2m)$

线性方程组

通过高斯消元可以解线性方程组

对于$i = 1, 2, \cdots, n$:

- 找到第i列的一个非0元素$a_{j,i}$（为避免精度误差，最好选绝对值最大的）
- 把第j行交换到第i行
- 对于$j = i + 1 \cdots n$，
  令$\forall k, a_{j,k} \leftarrow a_{j,k} - a_{i,k} \cdot \frac{a_{j,i}}{ai,i}$，$b_j \leftarrow b_j - b_i \cdot \frac{a_{j,i}}{ai,i}$
- 得到上三角矩阵，回代得到解(x_i)

注意也有可能找不到非0元素，这样消出的矩阵有一些空行，如果某个空行的b不为0则无解

否则如果$n = m$则有唯一解

复杂度$O(n^2 m)$

线性方程组

通过高斯消元可以解线性方程组

对于$i = 1, 2, \cdots, n$:

- 找到第i列的一个非0元素$a_{j,i}$（为避免精度误差，最好选绝对值最大的）
- 把第j行交换到第i行
- 对于$j = i + 1 \cdots n$，
  令$\forall k, a_{j,k} \leftarrow a_{j,k} - a_{i,k} \cdot \frac{a_{j,i}}{a_{i,i}}$，$b_j \leftarrow b_j - b_i \cdot \frac{a_{j,i}}{a_{i,i}}$
- 得到上三角矩阵，回代得到解(x_i)

注意也有可能找不到非0元素，这样消出的矩阵有一些空行，如果某个空行的b不为0则无解

否则如果$n = m$则有唯一解

复杂度$O(n^2m)$

线性方程组

通过高斯消元可以解线性方程组

对于$i = 1, 2, \cdots, n$:

- 找到第i列的一个非0元素$a_{j,i}$（为避免精度误差，最好选绝对值最大的）
- 把第j行交换到第i行
- 对于$j = i + 1 \cdots n$，
  令$\forall k, a_{j,k} \leftarrow a_{j,k} - a_{i,k} \cdot \frac{a_{j,i}}{ai,i}$，$b_j \leftarrow b_j - b_i \cdot \frac{a_{j,i}}{ai,i}$
- 得到上三角矩阵，回代得到解(x_i)

注意也有可能找不到非0元素，这样消出的矩阵有一些空行，如果某个空行的b不为0则无解

否则如果$n = m$则有唯一解

复杂度$O(n^2 m)$

递推与矩阵乘法

矩阵乘法可以表示递推关系

例如Fibonacci数列$F_0 = F_1 = 1, F_n = F_{n-1} + F_{n-2}$，有

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

用快速幂优化矩阵乘法可以$O(\log n)$求Fibonacci数列的某一项

递推与矩阵乘法

矩阵乘法可以表示递推关系

例如Fibonacci数列$F_0 = F_1 = 1, F_n = F_{n-1} + F_{n-2}$，有

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

用快速幂优化矩阵乘法可以$O(\log n)$求Fibonacci数列的某一项

递推与矩阵乘法

矩阵乘法可以表示递推关系

例如Fibonacci数列$F_0 = F_1 = 1, F_n = F_{n-1} + F_{n-2}$，有

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

用快速幂优化矩阵乘法可以$O(\log n)$求Fibonacci数列的某一项

例：Codeforces 718C

给一个数列$a_1, \Psi a_2, \Psi \ldots, \Psi a_n$，你需要进行$m$次操作，操作有两种：

- 1 l r x——将所有$l \le i \le r$的a_i增加x
- 2 l r——计算$\sum_{i=l}^{r} F_{a_i} \bmod (10^9+7)$，$F_i$为Fibonaaci数列第$i$项

例：Codeforces 718C

- 操作1相当于区间乘矩阵
- 操作2相当于区间矩阵求和
- 线段树维护矩阵即可

复杂度$O(m\log n)$

线性递推

给定$a_1, a_2, \cdots, a_k$和$f_0, f_1, \cdots, f_{k-1}$，设递推数列$\{f_n\}$满足对于$i \geq k$时$f_i = \sum_{j=1}^{k} a_j f_{i-j}$，求$f_n$

- 矩阵快速幂$O(k^3 \log n)$
- 有没有更快的算法?

线性递推

给定$a_1, a_2, \cdots, a_k$和$f_0, f_1, \cdots, f_{k-1}$，设递推数列$\{f_n\}$满足对于$i \geq k$时$f_i = \sum_{j=1}^{k} a_j f_{i-j}$，求$f_n$

- 矩阵快速幂$O(k^3 \log n)$
- 有没有更快的算法?

线性递推

给定$a_1, a_2, \cdots, a_k$和$f_0, f_1, \cdots, f_{k-1}$，设递推数列$\{f_n\}$满足对于$i \geq k$时$f_i = \sum_{j=1}^{k} a_j f_{i-j}$，求$f_n$

- 矩阵快速幂$O(k^3 \log n)$
- 有没更快的算法?

线性递推

f_n可以表示为$f_0, f_1, \cdots, f_{k-1}$的一个线性组合

根据f_n的线性表示，可以$O(k^2 \log n)$推出f_{n+1}的线性表示以及f_{2n}的线性表示

于是就能在$O(k^2 \log n)$时间内求出解了

线性递推

f_n可以表示为$f_0, f_1, \cdots, f_{k-1}$的一个线性组合

根据f_n的线性表示，可以$O(k^2 \log n)$推出f_{n+1}的线性表示以及f_{2n}的线性表示

于是就能在$O(k^2 \log n)$时间内求出解了

线性递推

f_n可以表示为$f_0, f_1, \cdots, f_{k-1}$的一个线性组合

根据f_n的线性表示，可以$O(k^2 \log n)$推出f_{n+1}的线性表示以及f_{2n}的线性表示

于是就能在$O(k^2 \log n)$时间内求出解了

例：NOI2017 泳池

一个底边长为N、高为1001的01矩阵，每个位置为1的概率为q，0的概率为$1-q$

求紧贴下边界的最大全1子矩形的大小恰好等于K的概率模998244353

$N \le 10^9$，$K \le 1000$

例：NOI2017 泳池

一个底边长为N、高为1001的01矩阵，每个位置为1的概率为q，0的概率为$1-q$

求紧贴下边界的最大全1子矩形的大小恰好等于K的概率模998244353

$N \le 10^9$，$K \le 1000$

例：NOI2017 泳池

一个底边长为N、高为1001的01矩阵，每个位置为1的概率为q，0的概率为$1-q$

求紧贴下边界的最大全1子矩形的大小恰好等于K的概率模998244353

$N \leq 10^9$，$K \leq 1000$

例：NOI2017 泳池

- 首先高度1001可以认为无限
- 设紧贴下边界的最大全1子矩形大小为S，则$P(S=K)=P(S\leq K)-P(S\leq K-1)$，因此问题转为求$S\leq K$的概率

例：NOI2017 泳池

- 首先高度1001可以认为无限
- 设紧贴下边界的最大全1子矩形大小为S，则$P(S=K)=P(S\le K)-P(S\le K-1)$，因此问题转为求$S\le K$的概率

例：NOI2017 泳池

DP，$f(i,j)$表示长度为i的矩阵，前j行均为1且紧贴下边界的最大全1子矩形不大于K的概率，如果$j \le \lfloor \frac{K}{i} \rfloor$，那么

$$f(i,j) = f(i,j+1) + \sum_{k=1}^{i} f(k-1,j+1)f(i-k,j)$$

否则$f(i,j) = 0$

注意到只有形如$\lfloor \frac{K}{i} \rfloor$的$j$是有用的，所以：

- 当$i \le K$时，总状态数为$O(K^2(1+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{K^2})) = O(K^2)$
- 当$i > K$时，$j=0$，可以转为线性递推，用上述方法可以做到$O(K^2 \log N)$

事实上还可以FFT优化到$O(K \log K \log N)$，但这题没有必要

例：NOI2017 泳池

DP，$f(i,j)$表示长度为i的矩阵，前j行均为1且紧贴下边界的最大全1子矩形不大于K的概率，如果$j \le \lfloor \frac{K}{i} \rfloor$，那么

$$f(i,j) = f(i,j+1) + \sum_{k=1}^{i} f(k-1,j+1)f(i-k,j)$$

否则$f(i,j) = 0$

注意到只有形如$\lfloor \frac{K}{i} \rfloor$的$j$是有用的，所以：

- 当$i \le K$时，总状态数为$O(K^2(1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{K^2})) = O(K^2)$
- 当$i > K$时，$j = 0$，可以转为线性递推，用上述方法可以做到$O(K^2 \log N)$

事实上还可以FFT优化到$O(K \log K \log N)$，但这题没有必要

例：NOI2017 泳池

DP，$f(i,j)$表示长度为i的矩阵，前j行均为1且紧贴下边界的最大全1子矩形不大于K的概率，如果$j \le \lfloor \frac{K}{i} \rfloor$，那么

$$f(i,j) = f(i,j+1) + \sum_{k=1}^{i} f(k-1,j+1)f(i-k,j)$$

否则$f(i,j) = 0$

注意到只有形如$\lfloor \frac{K}{i} \rfloor$的$j$是有用的，所以：

- 当$i \le K$时，总状态数为$O(K^2(1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{K^2})) = O(K^2)$
- 当$i > K$时，$j = 0$，可以转为线性递推，用上述方法可以做到$O(K^2 \log N)$

事实上还可以FFT优化到$O(K \log K \log N)$，但这题没有必要

例：NOI2017 泳池

DP，$f(i,j)$表示长度为i的矩阵，前j行均为1且紧贴下边界的最大全1子矩形不大于K的概率，如果$j \le \lfloor \frac{K}{i} \rfloor$，那么

$$f(i,j) = f(i,j+1) + \sum_{k=1}^{i} f(k-1,j+1)f(i-k,j)$$

否则$f(i,j) = 0$

注意到只有形如$\lfloor \frac{K}{i} \rfloor$的$j$是有用的，所以：

- 当$i \le K$时，总状态数为$O(K^2(1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{K^2})) = O(K^2)$
- 当$i > K$时，$j = 0$，可以转为线性递推，用上述方法可以做到$O(K^2 \log N)$

事实上还可以FFT优化到$O(K \log K \log N)$，但这题没有必要

例：NOI2017 泳池

DP，$f(i,j)$表示长度为i的矩阵，前j行均为1且紧贴下边界的最大全1子矩形不大于K的概率，如果$j \le \lfloor \frac{K}{i} \rfloor$，那么

$$f(i,j) = f(i,j+1) + \sum_{k=1}^{i} f(k-1,j+1)f(i-k,j)$$

否则$f(i,j) = 0$

注意到只有形如$\lfloor \frac{K}{i} \rfloor$的$j$是有用的，所以：

- 当$i \le K$时，总状态数为$O(K^2(1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{K^2})) = O(K^2)$
- 当$i > K$时，$j = 0$，可以转为线性递推，用上述方法可以做到$O(K^2 \log N)$

事实上还可以FFT优化到$O(K \log K \log N)$，但这题没有必要

End

祝大家下午AK!