基礎數論

日月卦長

判斷質數 $O(\sqrt{n})$

```
bool is_prime(int n) {
  if (n <= 1) return false;
  for (long long i = 2; i * i <= n; ++i)
    if (n % i == 0)
      return false;
  return true;
}
```

注意要特判以及使用 long long

- 如果正整數 n 是合數，則
  $$n = A \times B$$
- A, B 是大於 1 的正整數
- 顯然 $\min(A, B) \leq \sqrt{n}$
  - 否則會出現 $A \times B >$ n 的情況

經典題

- 輸入兩正整數 a, b
- 請你輸出 a~b 中有幾個質數

- $0 < a \leq b \leq 10^7$

κόσκινον Ἐρατοσθένους (埃拉托斯特尼篩法)

- 假定一開始，所有數字都是質數

2	3	4	5	6	7	8	9	10	11	12	13	14

κόσκινον Ἐρατοσθένους (埃拉托斯特尼篩法)

- 由前往後，如果遇到沒被刪除的數字，就把他的倍數刪掉

2	3	4	5	6	7	8	9	10	11	12	13	14

κόσκινον Ἐρατοσθένους (埃拉托斯特尼篩法)

- 下一個是 3

2	3	4	5	6	7	8	9	10	11	12	13	14

κόσκινον Ἐρατοσθένους (埃拉托斯特尼篩法)

- 下一個是 5

2	3	4	5	6	7	8	9	10	11	12	13	14

κόσκινον Ἐρατοσθένους (埃拉托斯特尼篩法)

- 下一個是 7

2	3	4	5	6	7	8	9	10	11	12	13	14

κόσκινον Ἐρατοσθένους (埃拉托斯特尼篩法)

- 下一個是 11

2	3	4	5	6	7	8	9	10	11	12	13	14

κόσκινον Ἐρατοσθένους (埃拉托斯特尼篩法)

- 下一個是 13

2	3	4	5	6	7	8	9	10	11	12	13	14

κόσκινον Ἐρατοσθένους (埃拉托斯特尼篩法)

- 剩下沒被刪掉的就是質數了

2	3	4	5	6	7	8	9	10	11	12	13	14

κόσκινον Ἐρατοσθένους (埃拉托斯特尼篩法)

```cpp
vector<bool> is_prime;
void sieve(int n) {
  is_prime.assign(n + 1, true);
  is_prime[0] = is_prime[1] = false;
  for (int i = 2; i <= n; ++i)
    if (is_prime[i])
      for (int j = i * 2; j <= n; j += i)
        is_prime[j] = false;
}
```

- 時間複雜度

$$
\begin{aligned}
O\left(n+\frac{n}{2}+\frac{n}{3}+\frac{n}{5}+\cdots\right) &= O\left(n+n\sum_{p}\frac{1}{p}\right) \\
&< O\left(n+n\sum_{x=1}^{\infty}\frac{1}{x}\right) \\
&= O(n\log n)
\end{aligned}
$$

- 實際上是

$$O(n\log\log n)$$

Mertens' second theorem

-

$$\lim_{n\to\infty}\left(\sum_{p\leq n}\left(\frac{1}{p}\right)-\ln\ln n-M\right)=0$$

$$M\approx 0.2614972128476427837554268386086958590516$$

常數優化

```
vector<bool> is_prime;
void sieve(int n) {
  is_prime.assign(n + 1, true);
  is_prime[0] = is_prime[1] = false;
  for (int i = 2; i * i <= n; ++i)
    if (is_prime[i])
      for (int j = i * i; j <= n; j += i)
        is_prime[j] = false;
}
```

- 設 x 是 $\sqrt{n}+1\sim n$ 之間的數字
- 若 x 是和數
  則它必然有一個質因數 $p \leq \sqrt{n}$
- 因此第一個迴圈只需要做到 $\sqrt{n}$

- 若 i 是質數，設數字 $x < i^2$
- 若 x 是 i 的倍數
  則 $\frac{x}{i}$ 必然有一個質因數 $p < i$
- 因此第二個迴圈可以從 i^2 開始計算

Sieve Of Euler 歐拉線性篩法 $O(n)$

- 每個合數只被自身最小的質因數篩選過一遍

```
vector<int> primes;
vector<bool> is_prime;
void sieve(int n) {
  primes.clear();
  is_prime.assign(n + 1, true);
  is_prime[0] = is_prime[1] = false;
  for (int i = 2; i < n; ++i) {
    if (is_prime[i]) primes.emplace_back(i);
    for (auto p : primes) {
      if (1LL * i * p > n) break;
      is_prime[i * p] = false;
      if (i % p == 0) break;
    }
  }
}
```

- 2×2
- $3 \times 2, 3 \times 3$
- 4×2
- $5 \times 2, 5 \times 3, 5 \times 5$
- $6 \times 2, 6 \times 3$
- $7 \times 2, 7 \times 3, 7 \times 5, 7 \times 7$
- 8×2

原問題的解法

```cpp
vector<int> primes;
int solve(int a, int b){
  auto L = lower_bound(primes.begin(), primes.end(), a);
  auto U = upper_bound(primes.begin(), primes.end(), b);
  return U - L;
}
```

利用二分搜 $O(\log n)$

建前綴和表 $O(n)$
每次查詢 $O(1)$

```cpp
vector<bool> is_prime;
vector<int> prefix_sum;
void init(){
  prefix_sum.assign(is_prime.begin(), is_prime.end());
  partial_sum(prefix_sum.begin(), prefix_sum.end(), prefix_sum.begin());
}
int solve(int a, int b) { return prefix_sum[b] - prefix_sum[a - 1]; }
```

篩法求 $1\sim n$ 每個數的質因數分解

- 透過埃式篩法可以建出空間大小為 $O(n\log\log n)$ 的表

對 x 做質因數分解的時間為 $O(\log x)$

```
vector<vector<int>> prime_factors(n + 1);
for (int i = 2; i <= n; ++i) {
  if (prime_factors[i].empty()) {
    for (int j = i; j <= n; j += i)
      prime_factors[j].emplace_back(i);
  }
}
```

$$12 = 2^2 \times 3^1$$

```
void print_prime_factor(int x) {
  int tmp = x;
  for (auto factor : prime_factors[x]) {
    int cnt = 0;
    for (; tmp % factor == 0; ++cnt)
      tmp /= factor;
    cout << factor << ": " << cnt << endl;
  }
}
```

篩法求 $1 \sim n$ 每個數的質因數分解

- 其實每個數字只需要紀錄 LPF (Least Prime Factor) 就夠了
- 變成可以用線性篩法求

對 x 做質因數分解的時間為 $O(\log x)$

```
vector<int> primes;
vector<int> LPFs(n + 1, 1);
for (int i = 2; i < n; ++i) {
  if (LPFs[i] == 1) {
    LPFs[i] = i;
    primes.emplace_back(i);
  }
  for (auto p : primes) {
    if (1LL * i * p > n) break;
    LPFs[i * p] = i;
    if (i % p == 0) break;
  }
}
```

```
void print_prime_factor(int x) {
  while (x != 1) {
    int factor = SPFs[x], cnt = 0;
    for (; x % factor == 0; ++cnt)
      x /= factor;
    cout << factor << ": " << cnt << endl;
  }
}
```

某種常見的題目需求

- Input:
  …

- Output:
  請你計算答案 ans 。
  但由於 ans 有可能非常大，所以請輸出 ans 除以 M 的餘數。

同餘關係

- 若整數 a, b 滿足 $a - b$ 整除於 m ，可以表示為 $a \equiv b \pmod{m}$

- $38 \equiv 14 \pmod{12}$
- $-8 \equiv 7 \pmod{5}$
- $-8 \equiv -3 \pmod{5}$

模運算

- 若 $a \equiv b \pmod{m}$, $p \equiv q \pmod{m}, c$ 為任意正數：

- $a + c \equiv b + c \pmod{m}$
- $ac \equiv bc \pmod{m}$
- $a^c \equiv b^c \pmod{m}$
- $a + p \equiv b + q \pmod{m}$
- $ap \equiv bq \pmod{m}$

利用 C++ 的模除 %

- 設整數 $a, b, m \geq 0$

如果希望結果不是負的只需要考慮上面這條

- $(a+b)\%m = (a\%m + b\%m)\%m$
- $(a-b)\%m = \begin{cases} (a\%m - b\%m + m)\%m, & a-b \geq 0 \\ (a\%m - b\%m + m)\%m - m, & a-b < 0 \end{cases}$
- $(a \times b)\%m = \big((a\%m) \times (b\%m)\big)\%m$
- $(a/b)\%m = ?$

模反元素

- 假設存在 $b \times b^{-1} \equiv 1 \pmod{m}$
- 則 $(a/b)\%m = (a \times b^{-1})\%m$
- 通常運算時原本的 (a/b) 會是整除

- 範例：
- $4 \times 3 \equiv 1 \pmod{11}$
- $(100/4)\%11 = (100 \times 3)\%11 = 300\%11 = 3$

費馬小定理

- 設 a 是一個整數，p 是任意質數，則 $a^p \equiv a \pmod p$

- $a^p \equiv a \pmod p$ ➔ $a^{p\text{-}1} \equiv 1 \pmod p$ ➔$a \times a^{p\text{-}2} \equiv 1 \pmod p$
- a 在模 p 情況下的模反元素是 $a^{p-2}\%p$
  - 用快速冪計算

快速冪 exponentiating by squaring

- 快速計算 a^b，由於數字可能很大通常會取除以 m 的餘數

```
unsigned fast_pow(unsigned a, unsigned b, unsigned m) {
  unsigned ans = 1;
  while (b) {
    if (b & 1) ans = 1ULL * (ans * a) % m;
    b >>= 1;
    a = 1ULL * (a * a) % m;
  }
  return ans;
}
```

費馬小定理證明

- 考慮 $\binom{p}{n} = \frac{p!}{n!(p-n)!}$，可知 $\binom{p}{n}$ 在 $n \neq p$ 且 $n \neq 0$ 時是 p 的倍數

$$
\begin{aligned}
(b+1)^p &\equiv \binom{p}{p} b^p + \binom{p}{p-1} b^{p-1} + \cdots + \binom{p}{0} b^0 \pmod{p} \\
&\equiv \binom{p}{p} b^p + \binom{p}{0} b^0 \pmod{p} \\
&\equiv b^p + 1 \pmod{p}
\end{aligned}
$$

費馬小定理證明

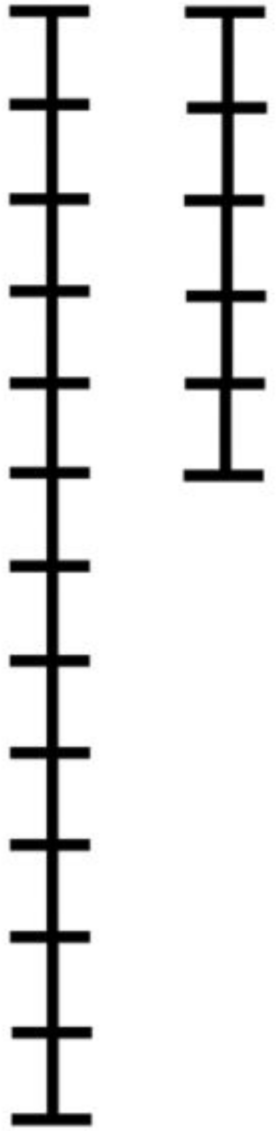
- $$\begin{aligned}(b+1)^p &\equiv b^p + 1 \pmod p \\ &\equiv \left((b-1)^p + 1\right) + 1 \pmod p \\ &\equiv \left((b-2)^p + 1\right) + 2 \pmod p \\ &\equiv \left((b-3)^p + 1\right) + 3 \pmod p \\ &\equiv b + 1 \pmod p\end{aligned}$$

- 令 $a = b + 1$，得到 $a^p \equiv a \pmod p$

輾轉相除法 Euclidean algorithm

- $\gcd(A, B) = \gcd(B, A\%B)$
  - Example: $\gcd(21,6) = \gcd(6,21\%6) = \gcd(6,3)$
- Code 就這樣，你可以去笑寫超長還爛掉的同學了
- C++17 內建 std::gcd

```
int gcd(int a, int b) {
  return !b ? a : gcd(b, a % b);
}
```



證明：$a\%b < \frac{a}{2}$

- 設 $r = a\%b$ ，則 $a = bk + r$

$$
\begin{aligned}
& a - bk < \frac{a}{2} \\
\leftrightarrow\ & bk > \frac{a}{2} \\
\leftrightarrow\ & bk > \frac{bk + r}{2} \\
\leftrightarrow\ & bk > r
\end{aligned}
$$

由於 $\gcd(a, b)$ 每次遞迴都會讓某個數字變少超過一半

因此時間複雜度是

$$O\left(\log \frac{\min(a, b)}{\gcd(a, b)}\right)$$

貝祖定理 – Extended Euclidean algorithm

- 對於整數 a, b, c，$ax + by = c$ 有整數解若且唯若 $\gcd(a, b) \mid c$。

```
pair<int, int> extgcd(int a, int b) {
  if (b == 0) return {1, 0};
  auto [xp, yp] = extgcd(b, a % b);
  return {yp, xp - a / b * yp};
}
```

$$
\begin{aligned}
ax + by &= \gcd(a, b) \\
&= \gcd(b, a\%b) \\
&= \gcd\left(b, a - \left\lfloor \frac{a}{b} \right\rfloor b\right) \\
&= bx' + \left(a - \left\lfloor \frac{a}{b} \right\rfloor b\right) y' \\
&= ay' + b\left(x' - \left\lfloor \frac{a}{b} \right\rfloor y'\right)
\end{aligned}
$$

貝祖等式 $ax + by = c$ 一般解

- 假設 $\gcd(a, b) \mid c$
- 設 $(x', y') = \text{extgcd}(a, b)$, $g = \gcd(a, b)$

$$\begin{aligned} ax' \qquad\qquad\quad &+ by' \qquad\qquad\quad = g \\ a\left(x' \times \frac{c}{g}\right) &+ b\left(y' \times \frac{c}{g}\right) = c \end{aligned}$$

- 因此 $x = \frac{x'c}{g}, y = \frac{y'c}{g}$ 是等式的其中一個解

貝祖等式 $ax + by = c$ 一般解

- 因此$x = \frac{x'c}{g}, y = \frac{y'c}{g}$ 是等式的其中一個解

$$
\begin{aligned}
& a\left(\frac{x'c}{g} + \frac{b}{g}\right) + b\left(\frac{y'c}{g} - \frac{a}{g}\right) \\
&= a\left(\frac{x'c}{g}\right) + b\left(\frac{y'c}{g}\right) + \frac{ab}{g} - \frac{ab}{g} \\
&= c
\end{aligned}
$$

- $x = \frac{x'c}{g} + \frac{b}{g}, y = \frac{y'c}{g} - \frac{a}{g}$ 也是等式的其中一個解

貝祖等式 $ax + by = c$ 一般解

- $x = \frac{x'c}{g} + \frac{b}{g}, y = \frac{y'c}{g} - \frac{a}{g}$ 也是等式的其中一個解
- 所以對於任意整數 k ：

$$x = \frac{x'c}{g} + k\frac{b}{g}$$
$$y = \frac{y'c}{g} - k\frac{a}{g}$$

- 是等式的一般解

Extgcd 求模反元素

- 我們想計算 a^{-1} 使得 $a \times a^{-1} \equiv 1 \pmod{m}$
- 若 $\boxed{\gcd(a,m) = 1}$，設 $(x', y') = \operatorname{extgcd}(a, m)$ ，則

$$\begin{aligned} ax' + my' &= 1 \\ ax' + my' &\equiv 1 \pmod{m} \\ ax' &\equiv 1 \pmod{m} \end{aligned}$$

- 因此 x' 就是 a 的模反元素
- 那 $\gcd(a,m) \neq 1$ 呢?

一次同餘方程 $ax \equiv b \pmod{m}$

- 證明： $ax \equiv b \pmod{m}$ 有解若且唯若 $\gcd(a, m) \mid b$

- 假設 $\gcd(a, m) \nmid b$
- 若 $ax \equiv b \pmod{m}$ 有解，存在 p, q 使得 $ap - b = mq$
- 移項得到 $b = ap - mq$，可知 b 是 $\gcd(a, m)$ 的倍數，矛盾

一次同餘方程 $ax \equiv b \pmod{m}$

- 證明： $ax \equiv b \pmod{m}$ 有解若且唯若 $\gcd(a, m) \mid b$

- 假設 $\gcd(a, m) \mid b$
- 設 $(x', y') = \text{extgcd}(a, m)$，則存在解 $x = x' \times \frac{b}{\gcd(a,m)}$

$$
\begin{aligned}
ax' \qquad\qquad\qquad\qquad &+ my' \qquad\qquad\qquad\qquad &= \gcd(a, m) \\
ax' \times \frac{b}{\gcd(a, m)} &+ my' \times \frac{b}{\gcd(a, m)} &= b
\end{aligned}
$$

一次同餘方程 $ax \equiv b \pmod{m}$ 一般解

- 證明： $ax \equiv b \pmod{m}$ 有解若且唯若 $\gcd(a, m) \mid b$

- 假設 $\gcd(a, m) \mid b$
- 設 $(x', y') = \mathrm{extgcd}(a, m)$，則對於任意整數 k 滿足

$$x = x' \times \frac{b}{\gcd(a, m)} \boxed{+ k \times \frac{m}{\gcd(a, m)}}$$

直接套用貝組等式的一般解公式

模反元素是否存在

- 對於 $b \times b^{-1} \equiv 1 \pmod{m}$
- 根據一次同餘方程的性質
- b^{-1}只有在 $\gcd(b, m) = 1$ 時存在

中國剩餘定理

有物不知其數
三三數之剩二
五五數之剩三
七七數之剩二
問物幾何

$$\begin{cases} x \equiv 2 \pmod 3 \\ x \equiv 3 \pmod 5 \\ x \equiv 2 \pmod 7 \end{cases}$$

$$x = ?$$

中國剩餘定理

- 若 $m_1, m_2, \ldots, m_n$ 兩兩互質 (定理中會用到模反元素)
- 則 $\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad \ldots \\ x \equiv a_n \pmod{m_n} \end{cases}$ 可以透過中國剩餘定理求解

中國剩餘定理

- 設 $M = m_1 \times m_2 \times \cdots \times m_n$ ， 並設 $M_i = \frac{M}{m_i}$
- 設 t_i 滿足 $M_i t_i \equiv 1 \pmod{m_i}$
- 對於任意整數 k ，存在方程式的一般解

$$
\begin{aligned}
x &= a_1 t_1 M_1 + a_2 t_2 M_2 + \cdots + a_n t_n M_n + kM \\
&= kM + \sum_{i=1}^{n} a_i t_i M_i
\end{aligned}
$$

中國剩餘定理 – 證明

- 考慮整數 $1 \le q \le n$

$$x = kM + \sum_{i=1}^{n} a_i t_i M_i$$

$$= \left(kM + \sum_{i=1, i \neq q}^{n} a_i t_i M_i \right) + a_q t_q M_q$$

這些都是 m_q 的倍數 $\quad t_q M_q \equiv 1 \pmod{m_q}$

排容原理

- 給你兩個正整數 $a, b(1 \leq a, b \leq 10^9)$
- 問你 $1,2,3, \dots, b$ 中，有多少數和 a 互質

想法

- 和 a 互質 ➔ $b-(1\sim b$ 中與 a 不互質的個數$)$
- 與 a 不互質 ➔ 有一個與 a 的共同質因數
  ➔ 任何是 a 的某個質因數的倍數都是答案 ➔ 對 a 做質因數分解

```cpp
vector<int> factorization(int n) {
  vector<int> ans;
  for (long long i = 2; i * i <= n; i++) {
    if (n % i == 0) {
      while (n % i == 0) n /= i;
      ans.emplace_back(i);
    }
  }
  if (n != 1) ans.emplace_back(n);
  return ans;
}
```

10^9 內每個數的質因數最多只有 9 個

範例 $a = 30, b = 15$

$a = 2 \times 3 \times 5$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

次數

0
1
2
3

- 與 a 不互質的數字數量：

範例 $a = 30, b = 15$

$a = 2 \times 3 \times 5$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

次數

0
1
2
3

- 與 a 不互質的數字數量：

$$\left\lfloor \frac{15}{2} \right\rfloor$$

範例 $a = 30, b = 15$

$a = 2 \times 3 \times 5$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

次數
0
1
2
3

- 與 a 不互質的數字數量：

$$\left\lfloor \frac{15}{2} \right\rfloor + \left\lfloor \frac{15}{3} \right\rfloor$$

範例 $a = 30, b = 15$

$a = 2 \times 3 \times 5$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

次數
0
1
2
3

- 與 a 不互質的數字數量：

$$\left\lfloor \frac{15}{2} \right\rfloor + \left\lfloor \frac{15}{3} \right\rfloor + \left\lfloor \frac{15}{5} \right\rfloor$$

範例 $a = 30, b = 15$

$a = 2 \times 3 \times 5$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

次數
0
1
2
3

- 與 a 不互質的數字數量：

$$\left\lfloor \frac{15}{2} \right\rfloor + \left\lfloor \frac{15}{3} \right\rfloor + \left\lfloor \frac{15}{5} \right\rfloor - \left(\left\lfloor \frac{15}{2 \times 3} \right\rfloor \right)$$

範例 $a = 30, b = 15$

$a = 2 \times 3 \times 5$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

次數
0
1
2
3

- 與 a 不互質的數字數量：

$$\left\lfloor\frac{15}{2}\right\rfloor + \left\lfloor\frac{15}{3}\right\rfloor + \left\lfloor\frac{15}{5}\right\rfloor - \left(\left\lfloor\frac{15}{2 \times 3}\right\rfloor + \left\lfloor\frac{15}{2 \times 5}\right\rfloor\right)$$

範例 $a = 30, b = 15$

$a = 2 \times 3 \times 5$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

次數

0
1
2
3

- 與 a 不互質的數字數量：

$$\left\lfloor\frac{15}{2}\right\rfloor + \left\lfloor\frac{15}{3}\right\rfloor + \left\lfloor\frac{15}{5}\right\rfloor - \left(\left\lfloor\frac{15}{2\times 3}\right\rfloor + \left\lfloor\frac{15}{2\times 5}\right\rfloor + \left\lfloor\frac{15}{3\times 5}\right\rfloor\right)$$

範例 $a = 30, b = 15$

$a = 2 \times 3 \times 5$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

次數

0
1
2
3

- 與 a 不互質的數字數量：

$$\left\lfloor\frac{15}{2}\right\rfloor + \left\lfloor\frac{15}{3}\right\rfloor + \left\lfloor\frac{15}{5}\right\rfloor - \left(\left\lfloor\frac{15}{2\times 3}\right\rfloor + \left\lfloor\frac{15}{2\times 5}\right\rfloor + \left\lfloor\frac{15}{3\times 5}\right\rfloor\right) + \left\lfloor\frac{15}{2\times 3\times 5}\right\rfloor = 11$$

$15 - 11 = 4$ 就是與 a 互質的個數

排容原理 – 關鍵程式碼

```
int a, b;
cin >> a >> b;
long long ans = 0; // 要注意計算過程中 overflow
auto factor = factorization(a);
unsigned S = (1u << factor.size()) - 1;
for (unsigned subset = 1; subset <= S; ++subset) {
  int flag = -1;
  long long product = 1;
  for (size_t i = 0; i < factor.size(); ++i) {
    if ((subset >> i) & 1) {
      flag *= -1;
      product *= factor[i];
    }
  }
  ans += flag * (b / product);
}
cout << b - ans << endl;
```

若 a 的質因數有 k 個
時間複雜度為 $O\left(\sqrt{a} + k \times 2^k\right)$

可以把 $b - ans$ 的操作放進迴圈中
當作回家作業自己思考一下