Algorithms for Programming Contests - Week 09

Prof. Dr. Javier Esparza
Pranav Ashok, A. R. Balasubramanian,
Tobias Meggendorfer, Philipp Meyer,
Mikhail Raskin,
conpra@in.tum.de

16. Juni 2020

Number Theory

Number Theory: *the study of integers*

- Around 1800 BC: Pythagorean triples in Mesopotamia
- Classical Greece (500-200 BC): Pythagoras, Plato, Euclid, Archimedes
- China (300-500 CE): Sun Tzu/Sunzi
- India (following centuries)
- Fibonacci (late 12th century)
- Early modern age: Fermat (17th), Euler (18th), Gauss (18/19th)

Number Theory

Subdivisions of Number Theory

- Elementary Tools
- Analytic Number Theory
- Algebraic Number Theory
- Diophantine Geometry
- Probabilistic Number Theory
- Arithmetic Combinatorics
- Computational/Algorithmic Number Theory

Basic terminology

- We study the set of integers $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$.
- Basic operations: addition $+$ and multiplication $\cdot$.
- Form an algebraic ring $(\mathbb{Z}, +, \cdot)$ with neutral elements 0 and 1.
- Non-negative integers: $\mathbb{Z}_{\geq 0} = \{0, 1, 2, \dots\}$.
- Positive integers: $\mathbb{Z}_{>0} = \{1, 2, \dots\}$.
- Prime numbers: $\mathbb{P}$.

Big Integers

- In C++ or Java, primitive data types cannot represent all integers:
  - C++: maxValue(`unsigned long long`) $= 2^{64} - 1 \approx 1.84 \cdot 10^{19}$
  - Java: maxValue(`long`) $= 2^{63} - 1 \approx 9.22 \cdot 10^{18}$
- For even larger integers use number system with base b:
  - Number $x = (x_n x_{n-1} \dots x_1 x_0)_b$ where $0 \leq x_i < b$
  - Value $\sum_{i=0}^{n} x_i \cdot b^i$

Big Integers

Addition

If $x = x_n \dots x_0$ and $y = y_n \dots y_0$, then $x + y = z = z_{n+1} z_n \dots z_n$ defined by:

$$c_i := \begin{cases} 1 & \text{if } i \geq 1 \text{ and } x_{i-1} + y_{i-1} \geq b \\ 0 & \text{otherwise} \end{cases}$$

$$z_i := \begin{cases} x_i + y_i + c_i & \text{if } x_i + y_i + c_i < b \\ x_i + y_i + c_i - b & \text{otherwise} \end{cases}$$

Big Integers

Multiplication (using long multiplication)

If $x = x_n \dots x_0$ and $y = y_m \dots y_0$, then

$$x \cdot y = \sum_{i=0}^{n} \sum_{j=0}^{m} x_i \cdot y_j \cdot b^{i+j}$$

- For product of digits, use hash tables or built-in operations.
- Additionally, keep track of sign when dealing with negative integers.
- Handle special cases.

Many more efficient algorithms available, e.g.: Toom-Cook multiplication, Schönhage-Strassen algorithm, Fast Fourier Transform.

Big Integers

- Choose base b so that invidiual digits fit into `long` or `int` datatypes.
- Space optimal: Base equal to the maximum value.
- Easier computation: Use only half the space to avoid overflows.
- Easier printing: Use $b = 10^k$ for some k.

Big Integers

- Choose base b so that invidiual digits fit into `long` or `int` datatypes.
- Space optimal: Base equal to the maximum value.
- Easier computation: Use only half the space to avoid overflows.
- Easier printing: Use $b = 10^k$ for some k.

- For Python: long arithmetics by default.
- For Java: use BigInteger class.
- For Julia: use BigInt type.
- For C++: not in standard library, write class yourself or use existing implementations or use another language.

Rational Numbers

Common problem with floating point numbers

- loss of significance
- rounding issues

Rational Numbers

Common problem with floating point numbers

- loss of significance
- rounding issues

Store numbers as rationals $\frac{a}{b}$ if exact calculations are required.

- Sum: $\frac{a}{b} + \frac{c}{d} = \frac{a \cdot d + b \cdot c}{b \cdot d}$
- Difference: $\frac{a}{b} - \frac{c}{d} = \frac{a \cdot d - b \cdot c}{b \cdot d}$
- Product: $\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$
- Quotient: $\frac{a}{b} \div \frac{c}{d} = \frac{a \cdot d}{b \cdot c}$
- Simplify rational number $\frac{a}{b}$ by canceling with $\gcd(a, b)$.
- Never divide by 0!

Fast Exponentiation

Exponentiation

For $x \in \mathbb{Z}$ and $n \in \mathbb{Z}_{\geq 0}$:

$$x^n = \underbrace{x \cdot x \cdot \ldots x \cdot x}_{n \text{ multiplications}}$$

More efficient: with $n = (n_k \ldots n_0)_2$, use

$$x^n = x^{(n_k \ldots n_0)_2} = x^{\sum_{i=0}^{k} n_i \cdot 2^i} = \prod_{i=0}^{k} x^{n_i \cdot 2^i} = \prod_{i=0}^{k} \left(x^{2^i}\right)^{n_i}$$

Use $x^0 = 1$, $x^1 = x$, $x^2 = x \cdot x$ and reuse results with $x^{2^i} = \left(x^{2^{i-1}}\right)^2$.
Only $\mathcal{O}(k) = \mathcal{O}(\log n)$ multiplications.

Fast Exponentiation Example

Naive Approach:

$$5^{19} = \underbrace{5 \cdot 5 \cdot \ldots 5 \cdot 5}_{\text{19 multiplications}}$$

Fast Exponentiation:

$$5^{19} = 5^{(10011)_2} = 5^{1+2+16} = 5^1 \cdot 5^2 \cdot 5^{16} = \left(5^{2^0}\right)^1 \cdot \left(5^{2^1}\right)^1 \cdot \left(5^{2^4}\right)^1$$

$$= \left(5^{2^4}\right)^1 \cdot \left(5^{2^3}\right)^0 \cdot \left(5^{2^2}\right)^0 \cdot \left(5^{2^1}\right)^1 \cdot \left(5^{2^0}\right)^1$$

Fast Exponentiation Example

Naive Approach:

$$5^{19} = \underbrace{5 \cdot 5 \cdot \ldots 5 \cdot 5}_{\text{19 multiplications}}$$

Fast Exponentiation:

$$5^{19} = 5^{(10011)_2} = 5^{1+2+16} = 5^1 \cdot 5^2 \cdot 5^{16} = \left(5^{2^0}\right)^1 \cdot \left(5^{2^1}\right)^1 \cdot \left(5^{2^4}\right)^1$$

$$= \left(5^{2^4}\right)^1 \cdot \left(5^{2^3}\right)^0 \cdot \left(5^{2^2}\right)^0 \cdot \left(5^{2^1}\right)^1 \cdot \left(5^{2^0}\right)^1$$

Or maybe:

$$5^{19} = 5^{9\times2+1} = 5^{(((1\times2)\times2)\times2+1)\times2+1} = \left(\left(\left(5^2\right)^2\right)^2 \times 5\right)^2 \times 5$$

Divisibility

Let $a, b \in \mathbb{Z}$. We say that *a divides b*, written as $a \mid b$, if there exists $k \in \mathbb{Z}$ such that $ak = b$.

- Note that $a \mid 0$ for any a, and $0 \mid b$ implies $b = 0$.
- If $a \mid b$ and $a \neq 0$, the k is uniquely determined. Then $k := \frac{b}{a}$.

Divisibility

Let $a, b \in \mathbb{Z}$. We say that a *divides* b, written as $a \mid b$, if there exists $k \in \mathbb{Z}$ such that $ak = b$.

- Note that $a \mid 0$ for any a, and $0 \mid b$ implies $b = 0$.
- If $a \mid b$ and $a \neq 0$, the k is uniquely determined. Then $k := \frac{b}{a}$.

An integer $p \in \mathbb{Z}_{>0}$ is a *prime number* if $p \neq 1$ and for all $k \in \mathbb{Z}_{>0}$, if $k \mid p$, then $k = 1$ or $k = p$.

Divisibility

Let $a, b \in \mathbb{Z}$. We say that a *divides* b, written as $a \mid b$, if there exists $k \in \mathbb{Z}$ such that $ak = b$.

- Note that $a \mid 0$ for any a, and $0 \mid b$ implies $b = 0$.
- If $a \mid b$ and $a \neq 0$, the k is uniquely determined. Then $k := \frac{b}{a}$.

An integer $p \in \mathbb{Z}_{>0}$ is a *prime number* if $p \neq 1$ and for all $k \in \mathbb{Z}_{>0}$, if $k \mid p$, then $k = 1$ or $k = p$.

Two integers $a, b \in \mathbb{Z}_{>0}$ are *coprime* if for all $k \in \mathbb{Z}_{>0}$, if $k \mid a$ and $k \mid b$, then $k = 1$.

Sieve of Eratosthenes

Algorithm 1 Sieve of Eratosthenes

Input: Integer n
Output: All prime numbers p with $p \leq n$.

```
procedure SIEVE(n)
    s[i] ← true for all i = 2, 3, ..., n.
    for i = 2, 3, ..., n do
        if s[i] = true then
            for j = 2i, 3i, 4i, ... with j ≤ n do
                s[j] ← false
            end for
        end if
    end for
    for i = 2, 3, ..., n with i[n] = true do
        output prime: i
    end for
end procedure
```

Sieve of Eratosthenes (optimized version)

Algorithm 2 Sieve of Eratosthenes

Input: Integer n
Output: All prime numbers p with $p \leq n$.

```
procedure SIEVE(n)
    s[i] ← true for all i = 2, 3, ..., n.
    for i = 2, 3, ..., ⌊√n⌋ do
        if s[i] = true then
            for j = i², i² + i, i² + 2i, ... with j ≤ n do
                s[j] ← false
            end for
        end if
    end for
    for i = 2, 3, ..., n with i[n] = true do
        output prime: i
    end for
end procedure
```

Analysis of Sieve of Eratosthenes

Running time

- Initialization of array $\mathcal{O}(n)$.
- Removing multiples $\sum_{p \leq n, p \in \mathbb{P}} \frac{n}{p} = n \sum_{p \leq n, p \in \mathbb{P}} \frac{1}{p} = \mathcal{O}(n \log \log n)$
- In total: $\mathcal{O}(n \log \log n)$

Euclidean Division

Lemma

Let $a, b \in \mathbb{Z}$ with $b \neq 0$. Then there exist unique integers $q, r \in \mathbb{Z}$ such that

$$a = bq + r \quad \textit{and} \quad 0 \leq r < |b|$$

We say that q is the quotient and r is the remainder of the Euclidean division of a and b, and define $a \operatorname{div} b := q$ and $a \bmod b := r$.

The values of $a \operatorname{div} b$ and $a \bmod b$ can be computed using long division.

Modular Arithmetic

Definition (Congruence modulo n)

Let $a, b \in \mathbb{Z}$ and $n \in \mathbb{Z}_{>0}$. We say that a and b are *congruent modulo n*, written as

$$a \equiv b \pmod{n}$$

if $n \mid a - b$, or, equivalently, if $a \bmod n = b \bmod n$.

Common rules for modular arithmetic:

- For a fixed n, the congruence is an equivalence relation.
- If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then

$$a + c \equiv b + d \pmod{n} \quad \text{and} \quad ac \equiv bd \pmod{n}$$

- For $p, q \in \mathbb{Z}_{>0}$ with p and q coprime, we have

$$a \equiv b \pmod{pq} \quad \text{iff} \quad a \equiv b \pmod{p} \text{ and } a \equiv b \pmod{q}$$

GCD & LCM

Let $a, b \in \mathbb{Z}$ with $a \neq 0$ or $b \neq 0$. The *greatest common divisor* of a and b is defined by:

$$\gcd(a, b) = \max\{k \in \mathbb{Z}_{>0} : (k \mid a) \wedge (k \mid b)\}$$

GCD & LCM

Let $a, b \in \mathbb{Z}$ with $a \neq 0$ or $b \neq 0$. The *greatest common divisor* of a and b is defined by:

$$\gcd(a, b) = \max\{k \in \mathbb{Z}_{>0} : (k \mid a) \wedge (k \mid b)\}$$

If $a \neq 0$ and $b \neq 0$, the *least common multiple* of a and b is defined by:

$$\operatorname{lcm}(a, b) = \min\{k \in \mathbb{Z}_{>0} : (a \mid k) \wedge (b \mid k)\}$$

GCD & LCM

Let $a, b \in \mathbb{Z}$ with $a \neq 0$ or $b \neq 0$. The *greatest common divisor* of a and b is defined by:

$$\gcd(a, b) = \max\{k \in \mathbb{Z}_{>0} : (k \mid a) \wedge (k \mid b)\}$$

If $a \neq 0$ and $b \neq 0$, the *least common multiple* of a and b is defined by:

$$\operatorname{lcm}(a, b) = \min\{k \in \mathbb{Z}_{>0} : (a \mid k) \wedge (b \mid k)\}$$

Properties of gcd and lcm:

- $\gcd(a, b) \cdot \operatorname{lcm}(a, b) = a \cdot b$.
- If $a \neq 0$, then $\gcd(0, a) = \gcd(a, 0) = a$.
- If $b \neq 0$, then $\gcd(a, b) = \gcd(b, a \bmod b)$.
- a and b are coprime iff $\gcd(a, b) = 1$.
- gcd of three numbers a, b, c can be computed as $\gcd(a, \gcd(b, c))$.

GCD & LCM

Consider the prime factorization of a and b, i.e.

$$a = \prod_{p_i \in \mathbb{P}} p_i^{r_i} \qquad b = \prod_{p_i \in \mathbb{P}} p_i^{s_i} \qquad \text{with } r_i, s_i \in \mathbb{Z}_{\geq 0}$$

The greatest common divisor and the least common multiple are then given by

$$\gcd(a, b) = \prod_{p_i \in \mathbb{P}} p_i^{\min\{r_i, s_i\}} \qquad \operatorname{lcm}(a, b) = \prod_{p_i \in \mathbb{P}} p_i^{\max\{r_i, s_i\}}$$

Note that

$$\gcd(a, b) \cdot \operatorname{lcm}(a, b) = \prod_{p_i \in \mathbb{P}} p_i^{\min\{r_i, s_i\} + \max\{r_i, s_i\}} = \prod_{p_i \in \mathbb{P}} p_i^{r_i + s_i} = a \cdot b$$

GCD & LCM - Example

$$
\begin{aligned}
a = &\quad 20 = 2^{2} \cdot 3^{0} \cdot 5^{1} \cdot 7^{0} \\
b = &\quad 42 = 2^{1} \cdot 3^{1} \cdot 5^{0} \cdot 7^{1} \\
\gcd(a, b) = &\quad\; 2 = 2^{1} \cdot 3^{0} \cdot 5^{0} \cdot 7^{0} \\
\operatorname{lcm}(a, b) = &\; 420 = 2^{2} \cdot 3^{1} \cdot 5^{1} \cdot 7^{1} \\
a \cdot b = &\; 840 = 2^{3} \cdot 3^{1} \cdot 5^{1} \cdot 7^{1}
\end{aligned}
$$

Euclidean Algorithm

Algorithm 3 Euclidean Algorithm

Input: Integers $a, b \in \mathbb{Z}$ with $a \neq 0$ or $b \neq 0$.
Output: Greatest common divisor of a and b.

```
procedure GCD(a, b)
    if b = 0 then
        return a
    else
        return GCD(b, a mod b)
    end if
end procedure
```

Complexity: Algorithm needs at most $\mathcal{O}(\log \min(a, b))$ steps. Total complexity defined by cost of mod operation.

Euclidean Algorithm - Example

Compute the greatest common divisor of 11 and 19:

$$
\begin{aligned}
\gcd(19, 11) &\longrightarrow 19 = 1 \cdot 11 + 8 \\
\gcd(11, 8) &\longrightarrow 11 = 1 \cdot 8 + 3 \\
\gcd(8, 3) &\longrightarrow 8 = 2 \cdot 3 + 2 \\
\gcd(3, 2) &\longrightarrow 3 = 1 \cdot 2 + 1 \\
\gcd(2, 1) &\longrightarrow 2 = 2 \cdot 1 + 0 \\
\gcd(1, 0) &= 1
\end{aligned}
$$

Bézout's Lemma

Lemma (Bézout's Lemma)

Let $a, b \in \mathbb{Z}_{>0}$ *and let* $d = \gcd(a, b)$*. Then there exist* $x, y \in \mathbb{Z}$ *such that*

$$ax + by = d \tag{1}$$

Additionally, there exist x, y *satisfying (1) with* $|x| \leq \frac{b}{d}$ *and* $|y| \leq \frac{a}{d}$*.*

Bézout's Lemma

Lemma (Bézout's Lemma)

Let $a, b \in \mathbb{Z}_{>0}$ and let $d = \gcd(a, b)$. Then there exist $x, y \in \mathbb{Z}$ such that

$$ax + by = d \tag{1}$$

Additionally, there exist x, y satisfying (1) with $|x| \leq \frac{b}{d}$ and $|y| \leq \frac{a}{d}$.

If $\gcd(a, b) = 1$, we also obtain the modular inverses:

$$ax \equiv 1 \pmod{b}$$
$$by \equiv 1 \pmod{a}$$

Modular Inverse

Example: Compute the modular inverse of 11 in group $(\mathbb{Z}_{19}, \cdot_{19})$, i.e. compute a number x such that

$$11 \cdot x \equiv 1 \pmod{19}.$$

Modular Inverse

Example: Compute the modular inverse of 11 in group $(\mathbb{Z}_{19}, \cdot_{19})$, i.e. compute a number x such that

$$11 \cdot x \equiv 1 \pmod{19}.$$

Compute $\gcd(19, 11)$:

$$\begin{aligned} 19 &= 1 \cdot 11 + 8 \\ 11 &= 1 \cdot 8 + 3 \\ 8 &= 2 \cdot 3 + 2 \\ 3 &= 1 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0 \end{aligned}$$

Modular Inverse

Example: Compute the modular inverse of 11 in group $(\mathbb{Z}_{19}, \cdot_{19})$, i.e. compute a number x such that

$$11 \cdot x \equiv 1 \pmod{19}.$$

Compute $\gcd(19, 11)$:

$$\begin{aligned} 19 &= 1 \cdot 11 + 8 \\ 11 &= 1 \cdot 8 + 3 \\ 8 &= 2 \cdot 3 + 2 \\ 3 &= 1 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0 \end{aligned}$$

Substitute:

$$\begin{aligned} 1 &= 3 - 1 \cdot 2 = 3 - 1 \cdot (8 - 2 \cdot 3) \\ &= -8 + 3 \cdot 3 = -8 + 3 \cdot (11 - 1 \cdot 8) \\ &= 3 \cdot 11 - 4 \cdot 8 = 3 \cdot 11 - 4 \cdot (19 - 1 \cdot 11) \\ &= -4 \cdot 19 + 7 \cdot 11 \end{aligned}$$

Modular Inverse

Example: Compute the modular inverse of 11 in group $(\mathbb{Z}_{19}, \cdot_{19})$, i.e. compute a number x such that

$$11 \cdot x \equiv 1 \pmod{19}.$$

Compute $\gcd(19, 11)$:

$$\begin{aligned} 19 &= 1 \cdot 11 + 8 \\ 11 &= 1 \cdot 8 + 3 \\ 8 &= 2 \cdot 3 + 2 \\ 3 &= 1 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0 \end{aligned}$$

Substitute:

$$\begin{aligned} 1 &= 3 - 1 \cdot 2 = 3 - 1 \cdot (8 - 2 \cdot 3) \\ &= -8 + 3 \cdot 3 = -8 + 3 \cdot (11 - 1 \cdot 8) \\ &= 3 \cdot 11 - 4 \cdot 8 = 3 \cdot 11 - 4 \cdot (19 - 1 \cdot 11) \\ &= -4 \cdot 19 + 7 \cdot 11 \end{aligned}$$

$$\begin{aligned} && 19 \cdot (-4) + 11 \cdot 7 &\equiv 1 \pmod{19} \\ \Leftrightarrow && 11 \cdot 7 &\equiv 1 \pmod{19} \end{aligned}$$

The modular inverse of 11 in $(\mathbb{Z}_{19}, \cdot_{19})$ is 7.

Extended Euclidean Algorithm

Algorithm 4 Euclidean Algorithm

Input: Integers $a, b \in \mathbb{Z}$ with $a \neq 0$ or $b \neq 0$.
Output: $\gcd(a, b)$ and integers x, y with $\gcd(a, b) = ax + by$.

```
procedure GCD(a, b)
    s ← 0, s' ← 1
    t ← 1, t' ← 0
    r ← b, r' ← a
    while r ≠ 0 do
        q ← r' div r
        (r', r) ← (r, r' − q · r)
        (s', s) ← (s, s' − q · s)
        (t', t) ← (t, t' − q · t)
    end while
    output gcd(a, b) = r'
    output (x, y) = (s', t')
end procedure
```

Extended Euclidean Algorithm

Algorithm 5 Euclidean Algorithm

Input: Integers $a, b \in \mathbb{Z}$ with $a \neq 0$ or $b \neq 0$.
Output: $\gcd(a, b)$ and integers x, y with $\gcd(a, b) = ax + by$.

procedure $\mathrm{GCD}(a, b)$
  $s \leftarrow 0, s' \leftarrow 1$
  $t \leftarrow 1, t' \leftarrow 0$
  $r \leftarrow b, r' \leftarrow a$
  while $r \neq 0$ **do**
    $q \leftarrow r'$ div r
    $(r', r) \leftarrow (r, r' - q \cdot r)$
    $(s', s) \leftarrow (s, s' - q \cdot s)$
    $(t', t) \leftarrow (t, t' - q \cdot t)$
  end while
  output $\gcd(a, b) = r'$
  output $(x, y) = (s', t')$
end procedure

i	r'	r	q	s'	s	t'	t
0	19	11		1	0	0	1
1	11	8	1	0	1	1	-1
2	8	3	1	1	-1	-1	2
3	3	2	2	-1	3	2	-5
4	2	1	1	3	-4	-5	7
5	1	0	2	**-4**	11	**7**	-19

$$\gcd(19, 11) = (-4) \cdot 19 + 7 \cdot 11$$

Chinese Remainder Theorem

Theorem (Chinese Remainder Theorem)

Let $n_1, \dots n_k \in \mathbb{Z}_{>0}$ be non-negative integers such that the n_i are pairwise coprime, and let $N := \prod_{i=1}^{k} n_i$. For integers $a_1, \dots a_k \in \mathbb{Z}$, define a set of congruences as follows:

$$x \equiv a_1 \pmod{n_1}$$
$$\vdots$$
$$x \equiv a_k \pmod{n_k}$$

Then

- *there exists an integer x satisfying all congruences, and*
- *if x and y satisfy all congruences, then $x \equiv y \pmod{N}$.*

Chinese Remainder Theorem (proof of uniqueness)

Proof (uniqueness modulo N).

Assume that x and y are solutions to the set of congruences. Then we have $x \equiv y \pmod{n_i}$ for all n_i. As the n_i are pairwise coprime, we obtain $x \equiv y \pmod{N}$.

Chinese Remainder Theorem (proof of uniqueness)

Proof (uniqueness modulo N).

Assume that x and y are solutions to the set of congruences. Then we have $x \equiv y \pmod{n_i}$ for all n_i. As the n_i are pairwise coprime, we obtain $x \equiv y \pmod{N}$.

- Consequently, in any interval of size N, there is exactly one solution.
- There is a unique solution in the interval $[0, N-1]$.

Chinese Remainder Theorem (proof of existence)

First consider the case with $k = 2$:

$$x \equiv a_1 \pmod{n_1}$$
$$x \equiv a_2 \pmod{n_2}$$

As $\gcd(n_1, n_2) = 1$, with Bézout's Lemma, we obtain m_1, m_2 such that

$$m_1 n_1 + m_2 n_2 = 1$$

Then

$$x = a_1 m_2 n_2 + a_2 m_1 n_1$$

is a solution, as

$$x = (a_1 m_2 n_2 + a_2 m_1 n_1) = a_1(1 - m_1 n_1) + a_2 m_1 n_1 = a_1 + (a_2 - a_1) m_1 n_1$$

Consider the case with $k > 2$:

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

Let $a_{1,2}$ be a solution to the first two congruences. Then the above and following set of congruences have the same the of solutions:

$$\begin{aligned} x &\equiv a_{1,2} \pmod{n_1 n_2} \\ x &\equiv a_3 \pmod{n_3} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$