date: 2025-11-12

# 1 solving congruences

imagine it is 10 o'clock. 5 hours later, it is 3 o'clock, not 15. this "resetting" behaviour is the essence of **modular arithmetic**.

## 1.1 arithmetic operations with $\mod n$

$\mathbb{Z}_m$ denotes the set of non-negative integers less than $m$ (the remainders when dividing by $m$).

modular arithmetic "preserves" addition and multiplication:

$$(a+b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$$
$$(a \cdot b) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$$

we define operations restricted to this set:

$$a +_m b = (a+b) \bmod m$$
$$a \cdot_m b = (a \cdot b) \bmod m$$

## 1.2 solving linear congruences

we look at equations of the form $ax \equiv b (\mod m)$.

### 1.2.1 additive inverses

look at an example: in order to solve $a + 8 \equiv 2 (\mod 11)$:

1. find the **additive inverse** of 8 modulo 11.
   - $-8 \equiv 3 (\mod 11)$
   - $-8 = (11 \cdot -1) + 3$
2. add 3 to both sides: $a \equiv 5 (\mod 11)$.

**note**: adding any multiple of $m$ preserves the value $(\mod m)$.

### 1.2.2 multiplicative inverses

to solve $ax \equiv b (\mod m)$ (a **linear congruence**), given values for $a$ and $b$, we need a value $\bar{a}$ such that $a \cdot \bar{a} \equiv 1 (\mod m)$. if we have this value, we can multiply it on both sides (similarly to multiplying by the reciprocal), and then simplify to find the solution. bézout's theorem, introduced below, helps to find that number.

**theorem** (bézout's theorem): if $\gcd(a, m) = 1$ ($a$ and $m$ are coprime), there exist integers $s$ and $t$ such that: $sa + tm = 1$.

this implies $sa \equiv 1 (\mod m)$, meaning $s$ is the multiplicative inverse of $a$.

**collary**: since $\gcd(a, m) = 1$ and $sa + tm = 1$, by substitution, we can get the **bézout identity**: $\gcd(a, b) = sa + tb$.

## 1.3 extended euclidean algorithm (eea)

the extended euclidean algorithm builds on the basic version that we have learned in the previous lecture. other than just finding the gcd, we also find the bézout numbers ($s, t$; refer to section above).

the first step of the extended euclidean algorithm is just by performing the basic version first; let's use $a = 99$ and $b = 78$ as an example.

$$
\begin{aligned}
99 &= 78 \cdot 1 + 21 \\
78 &= 21 \cdot 3 + 15 \\
21 &= 15 \cdot 1 + 6 \\
15 &= 6 \cdot 2 + 3 \\
6 &= 2 \cdot 3 + 0
\end{aligned}
$$

since the last remainder is zero(0), we disregard the last line and use the line before: $\gcd(99, 78) = 3$.

to find out the bézout numbers for the second step of the algorithm, we want to find how to express 3 with 99 and 78. to do that, we look at the previous work, start from the bottom, and work our way back.

$$
\begin{aligned}
3 &= 15 - (2 \cdot 6) && \\
6 &= 21 - (1 \cdot 15) && \\
3 &= 15 - (2 \cdot (21 - (1 \cdot 15))) && \text{(substitute 6)} \\
&= 15 - (2 \cdot 21) + (2 \cdot 15) && \text{(distribute -2)} \\
&= (3 \cdot 15) - (2 \cdot 21) && \text{(combine 15s, rearrange)} \\
15 &= 78 - (3 \cdot 21) && \\
3 &= (3 \cdot (78 - (3 \cdot 21))) - (2 \cdot 21) && \text{(substitute 15)} \\
&= (3 \cdot 78) - (9 \cdot 21) - (2 \cdot 21) && \text{(distribute 3)} \\
&= (3 \cdot 78) - (11 \cdot 21) && \text{(combine 21s)} \\
21 &= 99 - (1 \cdot 78) && \\
3 &= (3 \cdot 78) - (11 \cdot (99 - (1 \cdot 78))) && \text{(substitute 21)} \\
&= (3 \cdot 78) - (11 \cdot 99) + (11 \cdot 78) && \text{(distribute -11)} \\
3 &= \left(\boxed{14} \cdot 78\right) + \left(\boxed{-11} \cdot 99\right) && \text{(combine 78s)}
\end{aligned}
$$

we have found that $s = -11$ and $t = 14$ such that $3 = sa + tb$.

here's an iterative version of the algorithm in pseudocode:

algorithm: extended euclidean algorithm (eea)

```
1  procedure extended_euclid(a, b)
2  |  (old_r, r) := (a, b)
3  |  (old_s, s) := (1, 0)
4  |  (old_t, t) := (0, 1)
5  |  while r ≠ 0
6  |  |  q := ⌊old_r / r⌋
7  |  |  (old_r, r) := (r, old_r − q * r)
8  |  |  (old_s, s) := (s, old_s − q * s)
9  |  |  (old_t, t) := (t, old_t − q * t)
10 |  return (old_r, old_s, old_t)
```

## 1.4 chinese remainder theorem (crt)

used to solve systems of congruences where moduli are pairwise coprime.

assume we have a system like this:

$$\begin{aligned} x &\equiv a_1 (\mathrm{mod}\, m_1) \\ x &\equiv a_2 (\mathrm{mod}\, m_2) \\ &\cdots \\ x &\equiv a_n (\mathrm{mod}\, m_n) \end{aligned}$$

**solution**: let $m = m_1 m_2 ... m_n$. let $M_k = \frac{m}{m_k}$ (product of all moduli except $m_k$). let $y_k$ be the inverse of $M_k$ modulo $m_k$.

the unique solution modulo $m$ is:

$$x = \sum_{k=1}^{n} a_k M_k y_k$$

**example** (sunzi suanjing):

$$x \equiv 2 (\mathrm{mod}\, 3), x \equiv 3 (\mathrm{mod}\, 5), x \equiv 2 (\mathrm{mod}\, 7)$$

- $m = 3 \cdot 5 \cdot 7 = 105$.
- $M_1 = 35, y_1 = 2$ (since $35 \cdot 2 = 70 \equiv 1 \bmod 3$).
- $M_2 = 21, y_2 = 1$ (since $21 \cdot 1 = 21 \equiv 1 \bmod 5$).
- $M_3 = 15, y_3 = 1$ (since $15 \cdot 1 = 15 \equiv 1 \bmod 7$).
- $x = 2(35)(2) + 3(21)(1) + 2(15)(1) = 140 + 63 + 30 = 233 \equiv 23 (\mathrm{mod}\, 105)$.

## 1.5 advanced concepts

### 1.5.1 fermat's little theorem

if $p$ is prime and $p$ does not divide $a$:

$$a^{p-1} \equiv 1 (\mathrm{mod}\, p)$$

useful for computing large powers, e.g., $7^{222} \bmod 11$.

### 1.5.2 primitive roots & discrete log

- **primitive root ($r$)**: an element in $\mathbb{Z}_p$ whose powers generate all nonzero elements of $\mathbb{Z}_p$.
- **discrete log problem**: given $p, r, b$, find $e$ such that $r^e \equiv b (\mathrm{mod}\, p)$. there is no known polynomial time algorithm for this (foundation of cryptography).