CS70 – Spring 2026

Lecture 7 : Feb. 10

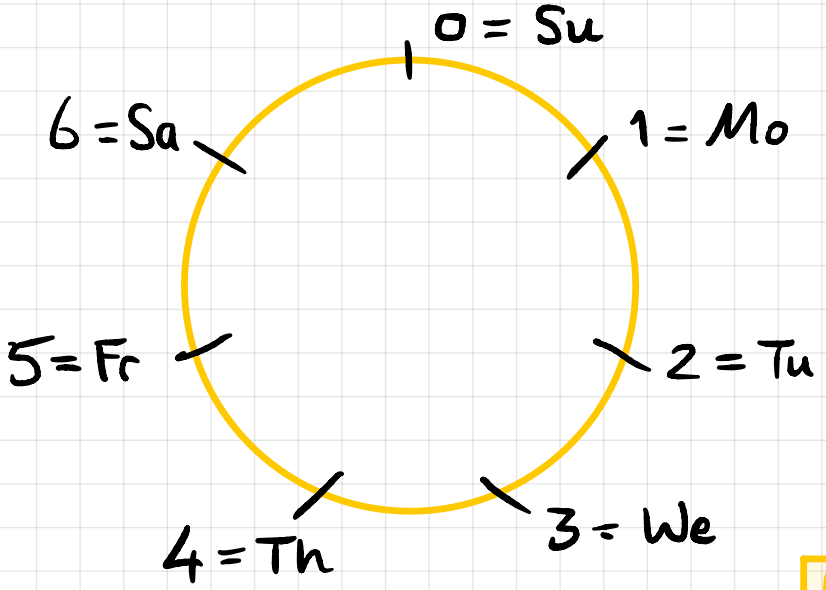
Next topic: Modular Arithmetic

- Modular arithmetic basics
- Computation mod n: addition/multiplication, exponentiation
- GCD and inverses
- Chinese Remainder Theorem
- Fermat's Little Theorem
- Application: RSA Cryptosystem

(More applications later)

Modular arithmetic : arithmetic limited to integers in a bounded range $\{0, 1, 2, \ldots, m-1\}$ for some m

E.g.: Days of the week $(m = 7)$



Today = Tu Feb. 10

Q: What day is Apr. 6 ?

A: # days = $18 + 31 + 6 = 55$

$= (7 \times 7) + 6$

Feb 10 : 2

$2 + 6 = 8 = (7 \times 1) + 1 \equiv 1$ Mo.

Residue classes (= equivalent numbers) mod 7

$0: \quad \{\ldots, -14, -7, 0, 7, 14, 21, \ldots\}$

$1: \quad \{\ldots, -13, -6, 1, 8, 15, 22, \ldots\}$

$2: \quad \{\ldots, -12, -5, 2, 9, 16, 23, \ldots\}$

$\vdots \qquad\qquad\qquad\qquad \vdots$

$6: \quad \{\ldots, -8, -1, 6, 13, 20, 27, \ldots\}$

We write (e.g.)

$23 \equiv 2 \pmod 7 \qquad -1 \equiv 6 \pmod 7$

$-14 \equiv 0 \pmod 7$

Generally: $\boxed{a \equiv b \pmod m \quad \text{iff} \quad m \mid (a-b)}$

Computation mod m

Example: $(759 \times 46) + 2268 \pmod 7$

$\equiv (3 \times 4) + 0 \pmod 7$

$\equiv 12 \pmod 7$

$\equiv \boxed{5} \pmod 7$

$[(k \times 7)+3] \times [(\ell \times 7)+4] \equiv 3 \times 4 \pmod 7$

Useful fact: Can always reduce intermediate results mod m $\longrightarrow$ no large numbers!

Formally:

$$(a \pm b) \equiv ((a \bmod m) \pm (b \bmod m)) \bmod m$$

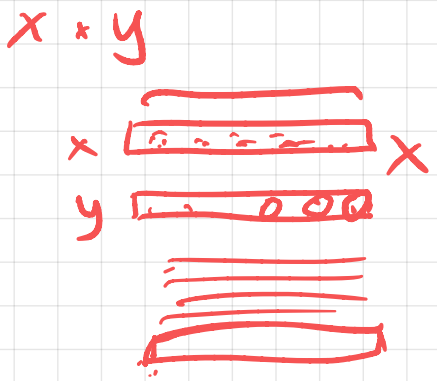
$$ab \equiv ((a \bmod m) \times (b \bmod m)) \bmod m$$

Proof: Ex. or see notes

Exponentiation

Addition & multiplication are efficient (at most $O(n^2)$ where n = # of bits in numbers)

What about exponentiation ?



Example : Compute $3^{42} \pmod{53}$

Method 1 : $3^2 = 9$, $3^3 = 27$, $3^4 = 81 \equiv 28$

$3^5 = 3 \times 28 = 84 \equiv 31$, ...

# of multiplications = 41

← generally, $m-1$ multiplicn's to compute x^m

Method 2 : [repeated squaring]

Note : # of digits in decimal number x is $\lfloor \log_{10} x \rfloor + 1$

# of bits in binary number x is $\lfloor \log_2 x \rfloor + 1$

Exponentiation

Example: Compute $3^{42} \pmod{53}$

Method 2: [repeated squaring]

$42 =$ (32 16 8 4 2 1) 101010 in binary

So $\boxed{3^{42} = 3^{32} \times 3^{8} \times 3^{2}}$

Compute:

$3^2 = 9$

$3^4 = 9^2 = 81 \equiv 28$

$3^8 = 28^2 = 784 \equiv 42$

$3^{16} = 42^2 = 1764 \equiv 15$

$3^{32} = 15^2 = 225 \equiv 13$

$3^{42} = 3^{32} \times 3^8 \times 3^2$

$= 13 \times 42 \times 9$

$\equiv 13 \times (-11) \times 9$

$\equiv 13 \times (-99)$

$\equiv 13 \times 7 = 91 \equiv \boxed{38}$

# of multiplications = 7 !

Generally:

# of multiplications to compute x^m

$\leq$ 2 × # of bits in m

$= O(\log_2 m)$

What about division?

Q: What does $4 \div 3 \pmod 7$ mean?

A: $4 \div 3 \equiv 4 \times (3^{-1}) \pmod 7$, where 3^{-1} is the inverse of 3 (mod 7)

Defn: x^{-1} (the inverse of x) mod m is the unique number (mod m) s.t.

$$x(x^{-1}) \equiv 1 \pmod m$$

ALERT: We are assuming here that x^{-1} exists and is unique! Need to prove this!

E.g.: $3 \times 5 = 15 \equiv 1 \pmod 7$

So 5 is the inverse of 3 (mod 7)

Example: Inverse of 12 mod 35?

$$3 \times 12 = 36 \equiv 1 \pmod{35}$$

So $12^{-1} = 3 \pmod{35}$

Example: Inverse of 10 mod 35?

$1 \times 10 = 10$
$2 \times 10 = 20$
$3 \times 10 = 30$
$4 \times 10 = 40 \equiv 5$
$5 \times 10 = 50 \equiv 15$
$6 \times 10 = 60 \equiv 25$
$7 \times 10 = 35 \equiv 0$
$8 \times 10 = 80 \equiv 10$
$9 \times 10 = 90 \equiv 20$
⋮ ⋮ (repeats)

10 has no inverse (mod 35)

← ALERT! Two non-zero numbers multiply to 0!

Theorem: x has an inverse mod m $\iff$ $\gcd(x,m)=1$

"x and m are coprime"

Proof: $\Rightarrow$ Sp. x has an inverse mod m

i.e. $ax \equiv 1 \pmod{m}$ for some a

So $ax = km + 1$ for some $k \in \mathbb{Z}$

Let $d = \gcd(x,m)$

Then $d \mid ax$ and $d \mid km$

Hence $d \mid (ax - km)$ i.e. $d \mid 1$

$\Rightarrow d = 1$ ✓

Theorem: x has an inverse mod m $\iff$ $\gcd(x,m)=1$

Proof: $\Leftarrow$ Consider the sequence of multiples of x:

$$0, x, 2x, \ldots, (m-1)x \pmod m$$

Claim: All of these numbers are different (mod m)

This finishes proof because one of the numbers in the seq. must be 1 (mod m)

Proof of Claim: Sp. for $\ast$ that $ax = bx \pmod m$ with $0 \le b < a \le m-1$

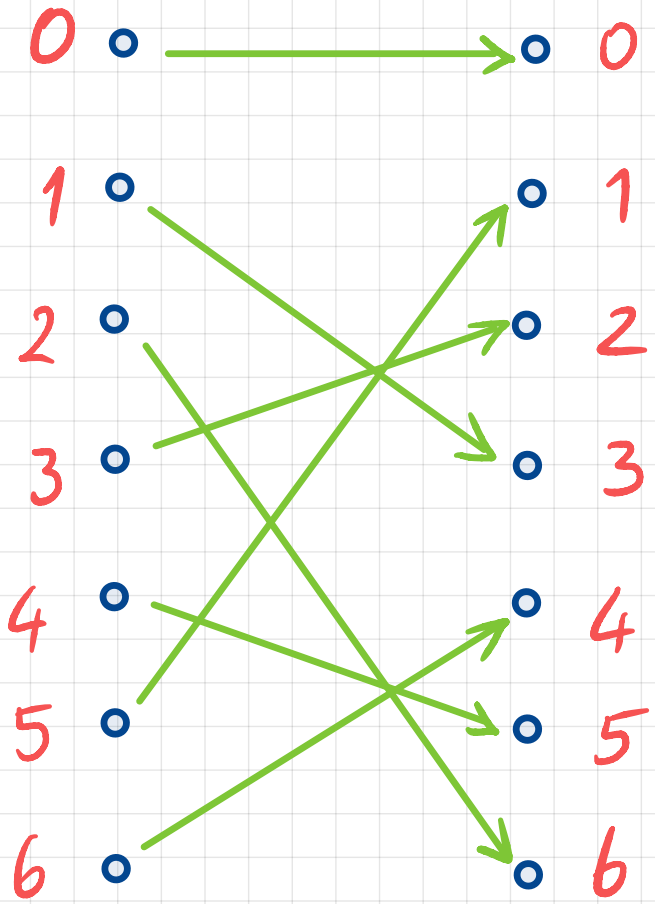
Then $m \mid (ax - bx) \Rightarrow m \mid x(a-b)$

But $\gcd(x,m)=1$, so $m \mid (a-b)$

Hence $a \equiv b \pmod m$ $\ast$ ✓

Example : $m = 7$, $x = 3$

Picture of the sequence $0, x, 2x, \ldots, 6x$



$3^{-1} = 5 \pmod 7$

bijection between $\{0, \ldots, m-1\}$ and itself

Corollary: Every $x \neq 0$ has an inverse (mod p) when p is a prime

Example: Inverses mod 11

$1^{-1} = 1$

$2^{-1} = 6$ $\quad 6^{-1} = 2$

$3^{-1} = 4$ $\quad 4^{-1} = 3$

$5^{-1} = 9$ $\quad 9^{-1} = 5$

$7^{-1} = 8$ $\quad 8^{-1} = 7$

$10^{-1} = 10$ $\quad [\equiv (-1) \bmod 11]$

Inverses mod 10

$1^{-1} = 1$

$3^{-1} = 7$ $\quad 7^{-1} = 3$

$9^{-1} = 9$

Integers coprime with 10: $\{1, 3, 7, 9\}$

Application of inverses

Solving linear equations mod m

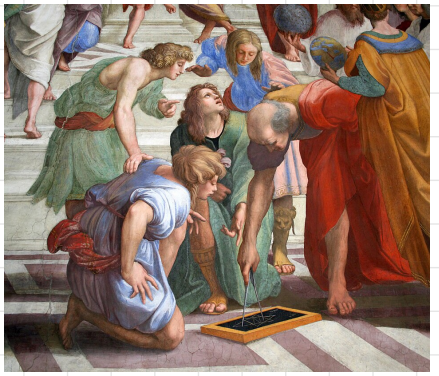
E.g.: $12x \equiv 11 \pmod{35}$

$12^{-1} \pmod{35} = 3$

multiply equn. by $12^{-1} = 3$:

$x \equiv 3 \times 11 = \boxed{33} \pmod{35}$

How to compute gcd ?



Euclid's Algorithm

Claim: If $x \geq y > 0$ then

$$\gcd(x,y) = \gcd(y, x \bmod y)$$

Proof: Suffices to prove that

$$d \mid x \;\&\; d \mid y \iff d \mid y \;\&\; d \mid x \bmod y \qquad (*)$$

$$x = (x \operatorname{div} y)\, y + (x \bmod y)$$

$(*)$ follows immediately from this $\square$

Euclid's Algorithm

```
function gcd(x, y)
    if y = 0 then output(x)
    else output (gcd(y, x mod y))
```

{assume $x \geq y \geq 0$ & $x > 0$ }

Correctness: strong induction on y (using previous Claim)

Examples:

gcd(35, 12)
= gcd(12, 11)
= gcd(11, 1)
= gcd(1, 0)
= 1 ✓

gcd(35, 10)
= gcd(10, 5)
= gcd(5, 0)
= 5 ✓

Euclid's Algorithm

```
function gcd(x, y)
    if y = 0 then output(x)
    else output (gcd(y, x mod y))
```

{assume $x \geq y \geq 0$ & $x > 0$}

Claim: Running time is $O(n)$ where n = # bits in x

Proof: Claim that after every two recursive calls, the first input (x) decreases by at least a factor of 2

Case(i): $y \leq \frac{x}{2}$ Then $x \longmapsto y \leq x/2$ ✓

Case(ii): $y > \frac{x}{2}$ Then $x \longmapsto y \longmapsto x \bmod y$

But $x \bmod y = x - y < x/2$ ✓

So # of bits in x argument decreases by 1 at most every 2 rec. calls. So total # of rec. calls is $O(n)$. □