Modular Arithmetic

Remember: $a \equiv b \pmod{m}$ means $a$ and $b$ have the same remainder when divided by $m$.

- Equivalently: $a \equiv b \pmod{m}$ iff $m \mid (a - b)$
- $a$ is congruent to $b$ mod $m$

**Theorem 7:** If $a_1 \equiv a_2 \pmod{m}$ and $b_1 \equiv b_2 \pmod{m}$, then

(a) $(a_1 + b_1) \equiv (a_2 + b_2) \pmod{m}$

(b) $a_1 b_1 \equiv a_2 b_2 \pmod{m}$

**Proof:** Suppose

- $a_1 = c_1 m + r, \ a_2 = c_2 m + r$
- $b_1 = d_1 m + r', \ b_2 = d_2 m + r'$

So

- $a_1 + b_1 = (c_1 + d_1)m + (r + r')$
- $a_2 + b_2 = (c_2 + d_2)m + (r + r')$

$m \mid ((a_1 + b_1) - (a_2 + b_2) = ((c_1 + d_1) - (c_2 + d_2))m$

- Conclusion: $a_1 + b_1 \equiv a_2 + b_2 \pmod{m}$. 
For multiplication:

- \(a_1b_1 = (c_1d_1m + r'c_1 + rd_1)m + rr'\)
- \(a_2b_2 = (c_2d_2m + r'c_2 + rd_2)m + rr'\)

\(m \mid (a_1b_1 - a_2b_2)\)

- Conclusion: \(a_1b_1 \equiv a_2b_2 \pmod{m}\).

**Bottom line:** addition and multiplication carry over to the modular world.

Modular arithmetic has lots of applications.

- Here are four . . .
Hashing

**Problem:** How can we efficiently store, retrieve, and delete records from a large database?

- For example, students records.

Assume, each record has a unique key

- E.g. student ID, Social Security #

Do we keep an array sorted by the key?

- Easy retrieval but difficult insertion and deletion.

How about a table with an entry for every possible key?

- Often infeasible, almost always wasteful.
- There are $10^{10}$ possible social security numbers.

Solution: store the records in an array of size $N$, where $N$ is somewhat bigger than the expected number of records.

- Store record with id $k$ in location $h(k)$
  - $h$ is the *hash function*
  - Basic hash function: $h(k) := k \pmod{N}$.
- A collision occurs when $h(k_1) = h(k_2)$ and $k_1 \neq k_2$.
  - Choose $N$ sufficiently large to minimize collisions
- Lots of techniques for dealing with collisions
Pseudorandom Sequences

For randomized algorithms we need a random number generator.

- Most languages provide you with a function “rand”.
- There is nothing random about rand!
  - It creates an apparently random sequence deterministically
  - These are called pseudorandom sequences

A standard technique for creating pseudorandom sequences: the linear congruential method.

- Choose a modulus $m \in N^+$,
- a multiplier $a \in \{2, 3, \ldots, m - 1\}$, and
- an increment $c \in Z_m = \{0, 1, \ldots, m - 1\}$.
- Choose a seed $x_0 \in Z_m$
  - Typically the time on some internal clock is used
- Compute $x_{n+1} = ax_n + c \pmod{m}$.

Warning: a poorly implemented rand, such as in C, can wreak havoc on Monte Carlo simulations.
ISBN Numbers

Since 1968, most published books have been assigned a 10-digit ISBN numbers:

• identifies country of publication, publisher, and book itself

• The ISBN number for DAM3 is 1-56881-166-7
All the information is encoded in the first 9 digits

• The 10th digit is used as a parity check

• If the digits are $a_1, \ldots, a_{10}$, then we must have
  
  \[ a_1 + 2a_2 + \cdots + 9a_9 + 10a_{10} \equiv 0 \pmod{11}. \]

• For DAM3, get
  
  \[ 1 + 2 \times 5 + 3 \times 6 + 4 \times 8 + 5 \times 8 + 6 \times 1 
  + 7 \times 1 + 8 \times 6 + 9 \times 6 + 10 \times 7 = 286 \equiv 0 \pmod{11} \]

• This test always detects errors in single digits and transposition errors
  
  ○ Two arbitrary errors may cancel out

Similar parity checks are used in universal product codes (UPC codes/bar codes) that appear on almost all items

• The numbers are encoded by thicknesses of bars, to make them machine readable
Casting out 9s

Notice that a number is equivalent to the sum of its digits mod 9. This can be used as a way of checking your addition. [More in class]
Linear Congruences

The equation $ax = b$ for $a, b \in R$ is uniquely solvable if $a \neq 0$: $x = ba^{-1}$.

- Can we also (uniquely) solve $ax \equiv b \pmod{m}$?
- If $x_0$ is a solution, then so is $x_0 + km \forall k \in \mathbb{Z}$
  - ...since $km \equiv 0 \pmod{m}$.

So, uniqueness can only be mod $m$.

But even mod $m$, there can be more than one solution:

- Consider $2x \equiv 2 \pmod{4}$
- Clearly $x \equiv 1 \pmod{4}$ is one solution
- But so is $x \equiv 3 \pmod{4}$!

**Theorem 8:** If $\gcd(a, m) = 1$ then there is a unique solution (mod $m$) to $ax \equiv b \pmod{m}$.

**Proof:** Suppose $r, s \in \mathbb{Z}$ both solve the equation:

- then $ar \equiv as \pmod{m}$, so $m \mid a(r - s)$
- Since $\gcd(a, m) = 1$, by Corollary 3, $m \mid (r - s)$
- But that means $r \equiv s \pmod{m}$

So if there’s a solution at all, then it’s unique mod $m$. 
Solving Linear Congruences

But why is there a solution to $ax \equiv b \pmod{m}$?

**Key idea:** find $a^{-1} \pmod{m}$; then $x \equiv ba^{-1} \pmod{m}$

- By Corollary 2, since $\gcd(a, m) = 1$, there exist $s, t$ such that
  
  $$as + mt = 1$$

- So $as \equiv 1 \pmod{m}$

- That means $s \equiv a^{-1} \pmod{m}$

- $x \equiv bs \pmod{m}$
The Chinese Remainder Theorem

Suppose we want to solve a system of linear congruences:

**Example:** Find $x$ such that

\[
\begin{align*}
x &\equiv 2 \pmod{3} \\
x &\equiv 3 \pmod{5} \\
x &\equiv 2 \pmod{7}
\end{align*}
\]

Can we solve for $x$? Is the answer unique?

**Definition:** $m_1, \ldots, m_n$ are pairwise relatively prime if each pair $m_i, m_j$ is relatively prime.

**Theorem 9 (Chinese Remainder Theorem):** Let $m_1, \ldots, m_n \in \mathbb{N}^+$ be pairwise relatively prime. The system

\[
x \equiv a_i \pmod{m_i} \quad i = 1, 2, \ldots, n \quad (1)
\]

has a unique solution modulo $M = \prod_i m_i$.

- The best we can hope for is uniqueness modulo $M$:
  - If $x$ is a solution then so is $x + kM$ for any $k \in \mathbb{Z}$.

**Proof:** First I show that there is a solution; then I’ll show it’s unique.
CRT: Existence

Key idea for existence:
Suppose we can find \( y_1, \ldots, y_n \) such that
\[
y_i \equiv a_i \pmod{m_i}
\]
\[
y_i \equiv 0 \pmod{m_j} \quad \text{if } j \neq i.
\]
Now consider \( y := \sum_{j=1}^{n} y_j \).
\[
\sum_{j=1}^{n} y_j \equiv a_i \pmod{m_i}
\]
• Since \( y_i = a_i \pmod{m_i} \) and \( y_j = 0 \pmod{m_j} \) if \( j \neq i \).
So \( y \) is a solution!

• Now we need to find \( y_1, \ldots, y_n \).
• Let \( M_i = M/m_i = m_1 \times \cdots \times m_{i-1} \times m_{i+1} \times \cdots \times m_n \).
• \( \gcd(M_i, m_i) = 1 \), since \( m_j \)'s pairwise relatively prime
  • No common prime factors among any of the \( m_j \)'s
Choose \( y_i' \) such that \( (M_i)y_i' \equiv a_i \pmod{m_i} \)
  • Can do that by Theorem 8, since \( \gcd(M_i, m_i) = 1 \).
Let \( y_i = y_i'M_i \).
  • \( y_i \) is a multiple of \( m_j \) if \( j \neq i \), so \( y_i \equiv 0 \pmod{m_j} \)
  • \( y_i = y_i'M_i \equiv a_i \pmod{m_i} \) by construction.
So \( y_1 + \cdots + y_n \) is a solution to the system, mod \( M \).
CRT: Example

Find $x$ such that
\[
    x \equiv 2 \pmod{3} \\
    x \equiv 3 \pmod{5} \\
    x \equiv 2 \pmod{7}
\]

Find $y_1$ such that $y_1 \equiv 2 \pmod{3}$, $y_1 \equiv 0 \pmod{5/7}$:
- $y_1$ has the form $y'_1 \times 5 \times 7$
- $35y'_1 \equiv 2 \pmod{3}$
- $y'_1 = 1$, so $y_1 = 35$.

Find $y_2$ such that $y_2 \equiv 3 \pmod{5}$, $y_2 \equiv 0 \pmod{3/7}$:
- $y_2$ has the form $y'_2 \times 3 \times 7$
- $21y'_2 \equiv 3 \pmod{5}$
- $y'_2 = 3$, so $y_2 = 63$.

Find $y_3$ such that $y_3 \equiv 2 \pmod{7}$, $y_3 \equiv 0 \pmod{3/5}$:
- $y_3$ has the form $y'_3 \times 3 \times 5$
- $15y'_3 \equiv 2 \pmod{7}$
- $y'_3 = 2$, so $y_3 = 30$.

Solution is $x = y_1 + y_2 + y_3 = 35 + 63 + 30 = 128$
CRT: Uniqueness

What if $x, y$ are both solutions to the equations?

- $x \equiv y \pmod{m_i} \Rightarrow m_i \mid (x - y)$, for $i = 1, \ldots, n$
- **Claim:** $M = m_1 \cdots m_n \mid (x - y)$
- so $x \equiv y \pmod{M}$

**Theorem 10:** If $m_1, \ldots, m_n$ are pairwise relatively prime and $m_i \mid b$ for $i = 1, \ldots, n$, then $m_1 \cdots m_n \mid b$.

**Proof:** By induction on $n$.

- For $n = 1$ the statement is trivial.

Suppose statement holds for $n = N$.

- Suppose $m_1, \ldots, m_{N+1}$ relatively prime, $m_i \mid b$ for $i = 1, \ldots, N + 1$.
- by IH, $m_1 \cdots m_N \mid b \Rightarrow b = m_1 \cdots m_N c$ for some $c$
- by assumption, $m_{N+1} \mid b$, so $m \mid (m_1 \cdots m_N)c$
- $\gcd(m_1 \cdots m_N, m_{N+1}) = 1$ (since $m_i$’s pairwise relatively prime $\Rightarrow$ no common factors)
- by Corollary 3, $m_{N+1} \mid c$

- so $c = dm_{N+1}$, $b = m_1 \cdots m_N m_{N+1}d$
- so $m_1 \cdots m_{N+1} \mid b$. 

12
An Application of CRT: Computer Arithmetic with Large Integers

Suppose we want to perform arithmetic operations (addition, multiplication) with extremely large integers

- too large to be represented easily in a computer

Idea:

- Step 1: Find suitable moduli $m_1, \ldots, m_n$ so that $m_i$’s are relatively prime and $m_1 \cdots m_n$ is bigger than the answer.

- Step 2: Perform all the operations mod $m_j$, $j = 1, \ldots, n$.
  - This means we’re working with much smaller numbers (no bigger than $m_j$)
  - The operations are much faster
  - Can do this in parallel

- Suppose the answer mod $m_j$ is $a_j$:
  - Use CRT to find $x$ such that $x \equiv a_j \pmod{m_j}$
  - The unique $x$ such that $0 < x < m_1 \cdots m_n$ is the answer to the original problem.
**Example:** The following are pairwise relatively prime:

\[ 2^{35} - 1, \ 2^{34} - 1, \ 2^{33} - 1, \ 2^{29} - 1, \ 2^{23} - 1 \]

We can add and multiply positive integers up to

\[(2^{35} - 1)(2^{34} - 1)(2^{33} - 1)(2^{29} - 1)(2^{23} - 1) > 2^{163}.\]
Fermat’s Little Theorem

Theorem 11 (Fermat’s Little Theorem):
(a) If $p$ prime and $\gcd(p, a) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.
(b) For all $a \in \mathbb{Z}$, $a^p \equiv a \pmod{p}$.

Proof. Let

$A = \{1, 2, \ldots, p - 1\}$

$B = \{1a \mod p, 2a \mod p, \ldots, (p-1)a \mod p\}$

Claim: $A = B$.

• $0 \notin B$, since $p \nmid ja$, so $B \subset A$.

• If $i \neq j$, then $ia \mod p \neq ja \mod p$

  • since $p \nmid (j - i)a$

Thus $|A| = p - 1$, so $A = B$.

Therefore,

$\prod_{i \in A} i \equiv \prod_{i \in B} i \pmod{p}$

$\Rightarrow (p - 1)! \equiv a(2a) \cdots (p - 1)a = (p - 1)! a^{p-1} \pmod{p}$

$\Rightarrow p \mid (a^{p-1} - 1)(p - 1)!$

$\Rightarrow p \mid (a^{p-1} - 1)$ [since $\gcd(p, (p - 1)!) = 1$]

$\Rightarrow a^{p-1} \equiv 1 \pmod{p}$

It follows that $a^p \equiv a \pmod{p}$

• This is true even if $\gcd(p, a) \neq 1$; i.e., if $p \mid a$

Why is this being taught in a CS course?
Private Key Cryptography

Alice (aka A) wants to send an encrypted message to Bob (aka B).

• A and B might share a private key known only to them.
• The same key serves for encryption and decryption.
• Example: Caesar’s cipher \( f(m) = m + 3 \mod 26 \) (shift each letter by three)
  ○ WKH EXWOHU GLG LW
  ○ THE BUTLER DID IT

This particular cryptosystem is very easy to solve

• Idea: look for common letters (E, A, T, S)
One Time Pads

Some private key systems are completely immune to cryptanalysis:

• A and B share the only two copies of a long list of random integers $s_i$ for $i = 1, \ldots, N$.

• A sends B the message $\{m_i\}_{i=1}^n$ encrypted as:

$$c_i = (m_i + s_i) \mod 26$$

• B decrypts A’s message by computing $c_i - s_i \mod 26$.

The good news: bulletproof cryptography
The bad news: horrible for e-commerce

• How do random users exchange the pad?
Public Key Cryptography

Idea of *public key cryptography* (Diffie-Hellman)

- Everyone’s encryption scheme is posted publically
  - e.g. in a “telephone book”
- If A wants to send an encoded message to B, she looks up B’s public key (i.e., B’s encryption algorithm) in the telephone book
- But only B has the decryption key corresponding to his public key

BIG advantage: A need not know nor trust B.

There seems to be a problem though:

- If we publish the encryption key, won’t everyone be able to decrypt?

Key observation: decrypting might be too hard, unless you know the key

- Computing $f^{-1}$ could be much harder than computing $f$

Now the problem is to find an appropriate $(f, f^{-1})$ pair for which this is true

- Number theory to the rescue
RSA: Key Generation

Generating encryption/decryption keys

- Choose two very large (hundreds of digits) primes $p, q$.
  - This is done using probabilistic primality testing
  - Choose a random large number and check if it is prime
  - By the prime number theorem, there are lots of primes out there
- Let $n = pq$.
- Choose $e \in \mathbb{N}$ relatively prime to $(p - 1)(q - 1)$.
  - How do you find $e$? Guess $e$, and use Euclid’s algorithm to check $\gcd(e, (p - 1)(q - 1)) = 1$
  - How many numbers less than $n$ are relatively prime to $(p - 1)(q - 1)$?
    - Lots: could choose $e$ to be another prime.
- Compute $d$, the inverse of $e$ modulo $(p - 1)(q - 1)$.
  - Can do this using using Euclidean algorithm
- Publish $n$ and $e$ (that’s your public key)
- Keep the decryption key $d$ to yourself.
RSA: Sending encrypted messages

How does someone send you a message?

- The message is divided into blocks each represented as a number \( M \) between 0 and \( n \). To encrypt \( M \), send

\[ C = M^e \mod n. \]

- Need to use fast exponentiation (\( 2 \log(n) \) multiplications) to do this efficiently

**Example:** Encrypt “stop” using \( e = 13 \) and \( n = 2537 \):

- \( s t o p \leftrightarrow 18 \ 19 \ 14 \ 15 \leftrightarrow 1819 \ 1415 \)
- \( 1819^{13} \mod 2537 = 2081 \) and
  \( 1415^{13} \mod 2537 = 2182 \) so
- \( 2081 \ 2182 \) is the encrypted message.
- We did not need to know \( p = 43, q = 59 \) for that.
RSA: Decryption

If you get an encrypted message $C = M^e \mod n$, how do you decrypt

- Compute $C^d \equiv M^{ed} \mod n$.
  - Can do this quickly using fast exponentiation again

**Claim:** $M^{ed} \equiv M \pmod{n}$

**Proof:** Since $ed \equiv 1 \pmod{(p-1)(q-1)}$

- $ed \equiv 1 \pmod{p-1}$ and $ed \equiv 1 \pmod{q-1}$

Since $ed = k(p-1) + 1$ for some $k$,

$$M^{ed} = (M^{p-1})^k M \equiv M \pmod{p}$$

(Fermat’s Little Theorem)

- True even if $p \mid M$

Similarly, $M^{ed} \equiv M \pmod{q}$

Since $p$, $q$, relatively prime, $M^{ed} \equiv M \pmod{n}$ (Theorem 10).

**Note:** Decryption would be easy for someone who can factor $N$.

- RSA depends on factoring being hard!
Digital Signatures

How can I send you a message in such a way that you’re convinced it came from me (and can convince others).

- Want an analogue of a “certified” signature

Cool observation:

- To send a message $M$, send $M^d \pmod{n}$
  
  - where $(n, e)$ is my public key

- Recipient (and anyone else) can compute $(M^d)^e \equiv M \pmod{n}$, since $M$ is public

- No one else could have sent this message, since no one else knows $d$. 
Probabilistic Primality Testing

RSA requires really large primes.

- This requires testing numbers for primality.
  - Although there are now polynomial tests, the standard approach now uses probabilistic primality tests.

Main idea in probabilistic primality testing algorithm:

- Choose $b$ between 1 and $n$ at random.
- Apply an easily computable (deterministic) test $T(b, n)$ such that
  - $T(b, n)$ is true (for all $b$) if $n$ is prime.
  - $T(b, n)$ there are lots of $b$’s for which $b$ is false if $n$ is not prime.

**Example:** Compute $\gcd(b, n)$.

- If $n$ is prime, $\gcd(b, n) = 1$
- If $n$ is composite, $\gcd(b, n) \neq 1$ for some $b$’s
  - Problem: there may not be that many witnesses.
Example: Compute $b^{n-1} \mod n$

- If $n$ is prime $b^{n-1} \equiv 1 \pmod{n}$ (Fermat)
- Unfortunately, there are some composite numbers $n$ such that $b^{n-1} \equiv 1 \pmod{n}$
  - These are called Carmichael numbers

There are tests $T(b, n)$ with the property that

- $T(b, n) = 1$ for all $b$ if $n$ is prime
- $T(b, n) = 0$ for at least $1/3$ of the $b$'s if $n$ is composite
- $T(b, n)$ is computable quickly (in polynomial time)

Constructing $T$ requires a little more number theory

- Beyond the scope of this course.

Given such a test $T$, it’s easy to construct a probabilistic primality test:

- Choose 100 (or 200) $b$’s at random
- Test $T(b, n)$ for each one
- If $T(b, n) = 0$ for any $b$, declare $b$ composite
  - This is definitely correct
- If $T(b, n) = 1$ for all $b$’s you chose, declare $n$ prime
  - This is highly likely to be correct
Prelim Coverage

• Chapter 0:
  • Sets
    • Operations: union, intersection, complementation, set difference
    • Proving equality of sets
  • Relations:
    • reflexive, symmetric, transitive, equivalence relations
    • transitive closure
  • Functions
    • Injective, surjective, bijective
    • Inverse function
  • Important functions and how to manipulate them:
    • exponent, logarithms, ceiling, floor, mod
  • Summation and product notation
  • Matrices (especially how to multiply them)
  • Proof and logic concepts
    • logical notions (⇒, ≡, ¬)
    • Proofs by contradiction
• Chapter 1
  ○ You don’t have to write algorithms in their notation
  ○ You may have to read algorithms in their notation

• Chapter 2
  ○ induction vs. strong induction
  ○ guessing the right inductive hypothesis
  ○ inductive (recursive) definitions

• Number Theory - everything we covered in class including
  ○ Fundamental Theorem of Arithmetic
  ○ gcd, lcm
  ○ Euclid’s Algorithm and its extended version
  ○ Modular arithmetic, linear congruences,
  ○ modular inverse and CRT
  ○ Fermat’s little theorem
  ○ RSA
  ○ Probabilistic primality testing

You need to know all the theorems and corollaries discussed in class.