CSE 311: Foundations of Computing

Topic 5: Number Theory
Applications of Predicate Logic

• Remainder of the course will use predicate logic to prove important properties of interesting objects
  – start with math objects that are widely used in CS
  – eventually more CS-specific objects

• Encode domain knowledge in predicate definitions

• Then apply predicate logic to infer useful results

<table>
<thead>
<tr>
<th>Domain of Discourse</th>
<th>Predicate Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integers</td>
<td>Even(x) ≡ ∃y (x = 2·y)</td>
</tr>
<tr>
<td></td>
<td>Odd(x) ≡ ∃y (x = 2·y + 1)</td>
</tr>
</tbody>
</table>
Mechanical vs Creative Predicate Logic

• We’ve done examples with “meaningless” predicates such as $\forall x P(x) \rightarrow \exists x P(x)$
  – Saw how to (often) *mechanically* solve by looking at “shape” of the goal.
  – We’ll need these skills in all domains!

• When we enter “interesting” domains of discourse, we will use domain knowledge.
  – We will see how to *creatively* solve goals, especially with rules like Intro $\lor$, Intro $\exists$, Elim $\land$, Elim $\forall$. 
Number Theory

• Direct relevance to computing
  – everything in a computer is a number
    colors on the screen are encoded as numbers

• Many significant applications in CS...
Pixels in Memory

- Memory is an array, so pixel positions must be mapped to array indexes

\[ 24 = 6 \times 4 \]
Pixels in Memory

6 x 4

pixel at (2, 4)

stored at index 16 = \(12 + 4\)
= \(2 \cdot 6 + 4\)
Pixels in Memory

6 x 4

Pixel at \((i, j)\)

Stored at index \(n\).

How do we calculate \(n\) from \(i\) and \(j\)?

\[ n = i \cdot 6 + j \]
Divisibility

Definition: “b divides a”

For $a, b$ with $b \neq 0$:

\[ b \mid a \iff \exists q \ (a = qb) \]

Check Your Understanding. Which of the following are true?

\[
\begin{array}{cccc}
5 & | & 1 & \quad 25 & | & 5 & \quad 5 & | & 0 & \quad 3 & | & 2 \\
1 & | & 5 & \quad 5 & | & 25 & \quad 0 & | & 5 & \quad 2 & | & 3
\end{array}
\]
Divisibility

**Definition: “b divides a”**

For $a, b$ with $b \neq 0$:

$b \mid a \iff \exists q (a = qb)$

**Check Your Understanding. Which of the following are true?**

- $5 \mid 1$: $5 \mid 1$ iff $1 = 5k$
- $25 \mid 5$: $25 \mid 5$ iff $5 = 25k$
- $5 \mid 0$: $5 \mid 0$ iff $0 = 5k$
- $3 \mid 2$: $3 \mid 2$ iff $2 = 3k$
- $1 \mid 5$: $1 \mid 5$ iff $5 = 1k$
- $5 \mid 25$: $5 \mid 25$ iff $25 = 5k$
- $0 \mid 5$: $0 \mid 5$ iff $5 = 0k$
- $2 \mid 3$: $2 \mid 3$ iff $3 = 2k$

**Domain of Discourse**

Integers
Recall: Elementary School Division

For $a, b$ with $b > 0$, we can divide $b$ into $a$.

If $b \mid a$, then, by definition, we have $a = qb$ for some $q$. The number $q$ is called the quotient.

Dividing both sides by $b$, we can write this as

$$
\frac{a}{b} = q
$$

(We want to stick to integers, though, so we’ll write $a = qb$.)

Recall: Elementary School Division

For $a, b$ with $b > 0$, we can divide $b$ into $a$.

If $b \nmid a$, then we end up with a remainder $r$ with $0 < r < b$.

Now,

instead of $\frac{a}{b} = q$ we have $\frac{a}{b} = q + \frac{r}{b}$

Multiplying both sides by $b$ gives us $a = qb + r$

(A bit nicer since it has no fractions.)
Recall: Elementary School Division

For $a, b$ with $b > 0$, we can divide $b$ into $a$.

If $b \mid a$, then we have $a = qb$ for some $q$.
If $b \nmid a$, then we have $a = qb + r$ for some $q, r$ with $0 < r < b$.

In general, we have $a = qb + r$ for some $q, r$ with $0 \leq r < b$, where $r = 0$ iff $b \mid a$. 
To put it another way, if we divide $b$ into $a$, we get a unique quotient $q = a \div b$ and non-negative remainder $r = a \mod b$. 

For $a, b$ with $b > 0$ there exist unique integers $q, r$ with $0 \leq r < b$ such that $a = qb + r$. 

Domain of Discourse

Integers
Pixels in Memory

6 x 4

pixel at \((i, j)\)

Stored at index \(n\).

How do we calculate \(n\) from \(i\) and \(j\)?

\[ n = i \cdot 6 + j \]
Pixels in Memory

6 x 4

0 1 2 3 4 5

0 1 2 3 4 5

i = n div 6
j = n mod 6
Number Theory

- Direct relevance to computing
  - important toolkit for programmers

- Many significant applications
  - Cryptography & Security
  - Data Structures
  - Distributed Systems
Modular Arithmetic
(and Its Applications)
Modular Arithmetic

- Arithmetic over a finite domain
- Almost all computation is over a finite domain
I’m ALIVE!

```java
public class Test {
    final static int SEC_IN_YEAR = 365*24*60*60;
    public static void main(String args[]) {
        System.out.println("I will be alive for at least " + SEC_IN_YEAR * 101 + " seconds.");
    }
}
```
public class Test {
    final static int SEC_IN_YEAR = 365*24*60*60;
    public static void main(String args[]) {
        System.out.println("I will be alive for at least " + SEC_IN_YEAR * 101 + " seconds.");
    }
}

---jGRASP exec: java Test
I will be alive for at least -186619904 seconds.

---jGRASP: operation complete.
Ordinary arithmetic

\[ 2 + 3 = 5 \]
Arithmetic on a Clock

\[2 + 3 = 5\]

\[23 = 3 \cdot 7 + 2\]

If \(a = 7q + r\), then \(r \equiv a \mod b\) is where you stop after taking \(a\) steps on the clock.
## Arithmetic, mod 7

\[ (a + b) \mod 7 \]

\[ (a \times b) \mod 7 \]
Modular Arithmetic

Definition: “a is congruent to b modulo m”

For $a, b, m$ with $m > 0$

$a \equiv_m b \iff m \mid (a - b)$

New notion of “sameness” that will help us understand modular arithmetic
Modular Arithmetic

**Definition: “a is congruent to b modulo m”**

For $a, b, m$ with $m > 0$

$$a \equiv_m b \iff m \mid (a - b)$$

The standard math notation is

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

A chain of equivalences is written

$$a \equiv b \equiv c \equiv d \pmod{m}$$

Many students find this confusing, so we will use $\equiv_m$ instead.
Modular Arithmetic

Definition: “a is congruent to b modulo m”

For \(a, b, m\) with \(m > 0\)

\[a \equiv_m b \iff m \mid (a - b)\]

Check Your Understanding. What do each of these mean? When are they true?

\(x \equiv_2 0\)

This statement is the same as saying “\(x\) is even”; so, any \(x\) that is even (including negative even numbers) will work.

\(-1 \equiv_5 19\)

This statement is true. \(19 - (-1) = 20\) which is divisible by 5

\(y \equiv_7 2\)

This statement is true for \(y\) in \{ ..., -12, -5, 2, 9, 16, ... \}. In other words, all \(y\) of the form \(2 + 7k\) for \(k\) an integer.
Modular Arithmetic: A Property

Let $a, b, m$ be integers with $m > 0$. Then, $a \equiv_m b$ if and only if $a \mod m = b \mod m$. 
Modular Arithmetic: A Property

Let $a, b, m$ be integers with $m > 0$. Then, $a \equiv_m b$ if and only if $a \mod m = b \mod m$.

Suppose that $a \mod m = b \mod m$.

By the division theorem, $a = mq + (a \mod m)$ and $b = ms + (b \mod m)$ for some integers $q, s$.

Goal: show $a \equiv_m b$, i.e., $m \mid (a - b)$. 
**Modular Arithmetic: A Property**

Let \( a, b, m \) be integers with \( m > 0 \). Then, \( a \equiv_m b \) if and only if \( a \mod m = b \mod m \).

Suppose that \( a \mod m = b \mod m \).

By the division theorem, \( a = mq + (a \mod m) \) and \( b = ms + (b \mod m) \) for some integers \( q,s \).

Then, \( a - b = (mq + (a \mod m)) - (ms + (b \mod m)) = m(q - s) + (a \mod m - b \mod m) = m(q - s) \) since \( a \mod m = b \mod m \)

**Goal:** show \( a \equiv_m b \), i.e., \( m \mid (a - b) \).
Modular Arithmetic: A Property

Let \( a, b, m \) be integers with \( m > 0 \).
Then, \( a \equiv_m b \) if and only if \( a \mod m = b \mod m \).

Suppose that \( a \mod m = b \mod m \).

By the division theorem, \( a = mq + (a \mod m) \) and 
\[ b = ms + (b \mod m) \] for some integers \( q, s \).

Then, 
\[ a - b = (mq + (a \mod m)) - (ms + (b \mod m)) \]
\[ = m(q - s) + (a \mod m - b \mod m) \]
\[ = m(q - s) \text{ since } a \mod m = b \mod m \]

Therefore, \( m \mid (a - b) \) and so \( a \equiv_m b \).
Modular Arithmetic: A Property

Let $a, b, m$ be integers with $m > 0$. Then, $a \equiv_m b$ if and only if $a \mod m = b \mod m$.

Suppose that $a \equiv_m b$.

Then, $m \mid (a - b)$ by definition of congruence. So, $a - b = km$ for some integer $k$ by definition of divides. Therefore, $a = b + km$. 
Modular Arithmetic: A Property

Let $a, b, m$ be integers with $m > 0$. Then, $a \equiv_m b$ if and only if $a \mod m = b \mod m$.

Suppose that $a \equiv_m b$.

Then, $m \mid (a - b)$ by definition of congruence. So, $a - b = km$ for some integer $k$ by definition of divides. Therefore, $a = b + km$.

By the Division Theorem, we have $a = qm + (a \mod m)$, where $0 \leq (a \mod m) < m$. 
Modular Arithmetic: A Property

Let $a, b, m$ be integers with $m > 0$. Then, $a \equiv_m b$ if and only if $a \mod m = b \mod m$.

Suppose that $a \equiv_m b$.

Then, $m \mid (a - b)$ by definition of congruence. So, $a - b = km$ for some integer $k$ by definition of divides. Therefore, $a = b + km$.

By the Division Theorem, we have $a = qm + (a \mod m)$, where $0 \leq (a \mod m) < m$.

Combining these, we have $qm + (a \mod m) = a = b + km$ or equiv., $b = qm - km + (a \mod m) = (q - k)m + (a \mod m)$. By the Division Theorem, we have $b \mod m = a \mod m$. 
The \textit{mod} \textit{m} function vs the \textit{\equiv}_m predicate

- What we have just shown
  - The \textit{mod} \textit{m} function maps any integer \textit{a} to a remainder \textit{a \mod m} \in \{0,1,\ldots,m-1\}.

  - Imagine grouping together all integers that have the same value of the \textit{mod} \textit{m} function
    That is, the same remainder in \{0,1,\ldots,m-1\}.

  - The \textit{\equiv}_m predicate compares integers \textit{a}, \textit{b}. It is true if and only if the \textit{mod} \textit{m} function has the same value on \textit{a} and on \textit{b}.
    That is, \textit{a} and \textit{b} are in the same group.
Recall: Familiar Properties of “=”

• If $a = b$ and $b = c$, then $a = c$.
  - i.e., if $a = b = c$, then $a = c$

• If $a = b$ and $c = d$, then $a + c = b + d$.
  - since $c = c$ is true, we can “$+$ $c$” to both sides

• If $a = b$ and $c = d$, then $ac = bd$.
  - since $c = c$ is true, we can “$\times c$” to both sides

These facts allow us to use algebra to solve problems
Recall: Properties of “=” Used in Algebra

| If \( a = b \) and \( b = c \), then \( a = c \). | “Transitivity” |
| If \( a = b \), then \( a + c = b + c \). | “Add Equations” |
| If \( a = b \), then \( ac = bc \). | “Multiply Equations” |

These are **Theorems** that we can use in proofs.

**Example:** given \( 5x + 4 = 2x + 25 \), prove that \( 3x = 21 \).

Let’s see how to do this in **formal** logic...
Recall: Properties of “=” Used in Algebra

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>If ( a = b ) and ( b = c ), then ( a = c ).</td>
<td>“Transitivity”</td>
</tr>
<tr>
<td>If ( a = b ), then ( a + c = b + c ).</td>
<td>“Add Equations”</td>
</tr>
<tr>
<td>If ( a = b ), then ( ac = bc ).</td>
<td>“Multiply Equations”</td>
</tr>
</tbody>
</table>

1. \( 5x + 4 = 2x + 25 \)                                          Given
2. \( -4 = -4 \)                                                  Algebra
3. \( 5x = 2x + 21 \)                                             Add Equations: 1, 2
4. \( -2x = -2x \)                                                Algebra
5. \( 3x = 21 \)                                                  Add Equations: 3, 4
Recall: Properties of “=” Used in Algebra

<table>
<thead>
<tr>
<th>If $a = b$ and $b = c$, then $a = c$.</th>
<th>“Transitivity”</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $a = b$, then $a + c = b + c$.</td>
<td>“Add Equations”</td>
</tr>
<tr>
<td>If $a = b$, then $ac = bc$.</td>
<td>“Multiply Equations”</td>
</tr>
</tbody>
</table>

1. $5x + 4 = 2x + 25$  
   Given

...  

5. $3x = 21$  
   Transitivity

**Careful:** prove $5x + 4 = 2x + 25 \Rightarrow 3x = 21$

**Not** $3x = 21 \Rightarrow 5x + 4 = 2x + 25$

the second is a “backward” proof
Recall: Familiar Properties of “=”

- If $a = b$ and $b = c$, then $a = c$.  
  - i.e., if $a = b = c$, then $a = c$

- If $a = b$ and $c = d$, then $a + c = b + d$.  
  - since $c = c$ is true, we can “$+ c$” to both sides

- If $a = b$ and $c = d$, then $ac = bd$.  
  - since $c = c$ is true, we can “$\times c$” to both sides

Same facts apply to “≤” with non-negative numbers

What about “≡ₘ”?
Let $m$ be a positive integer. If $a \equiv_m b$ and $b \equiv_m c$, then $a \equiv_m c$. 
Modular Arithmetic: Basic Property

Let $m$ be a positive integer. If $a \equiv_m b$ and $b \equiv_m c$, then $a \equiv_m c$.

Suppose that $a \equiv_m b$ and $b \equiv_m c$. 

Let $m$ be a positive integer. If $a \equiv_m b$ and $b \equiv_m c$, then $a \equiv_m c$.

Suppose that $a \equiv_m b$ and $b \equiv_m c$. Then, by the previous property, we have $a \mod m = b \mod m$ and $b \mod m = c \mod m$.

Putting these together, we have $a \mod m = c \mod m$, which says that $a \equiv_m c$, by the previous property.
Let $m$ be a positive integer. If $a \equiv_m b$ and $c \equiv_m d$, then $a + c \equiv_m b + d$. 
Modular Arithmetic: Addition Property

Let $m$ be a positive integer. If $a \equiv_m b$ and $c \equiv_m d$, then $a + c \equiv_m b + d$.

Suppose that $a \equiv_m b$ and $c \equiv_m d$. 
Modular Arithmetic: Addition Property

Let \( m \) be a positive integer. If \( a \equiv_m b \) and \( c \equiv_m d \), then \( a + c \equiv_m b + d \).

Suppose that \( a \equiv_m b \) and \( c \equiv_m d \). Unrolling the definitions, we can see that \( a - b = km \) and \( c - d = jm \) for some \( k, j \in \mathbb{Z} \).

Adding the equations together gives us
\[
(a + c) - (b + d) = m(k + j).
\]

By the definition of congruence, we have \( a + c \equiv_m b + d \).
Let $m$ be a positive integer. If $a \equiv_m b$ and $c \equiv_m d$, then $ac \equiv_m bd$. 
Let $m$ be a positive integer. If $a \equiv_m b$ and $c \equiv_m d$, then $ac \equiv_m bd$. 

Suppose that $a \equiv_m b$ and $c \equiv_m d$. 

---

**Modular Arithmetic: Multiplication Property**

Suppose that $a \equiv_m b$ and $c \equiv_m d$. 

Then $ac \equiv_m bd$. 

---

**Modular Arithmetic: Multiplication Property**

Suppose that $a \equiv_m b$ and $c \equiv_m d$. 

Then $ac \equiv_m bd$. 

---
Modular Arithmetic: Multiplication Property

Let $m$ be a positive integer. If $a \equiv_m b$ and $c \equiv_m d$, then $ac \equiv_m bd$.

Suppose that $a \equiv_m b$ and $c \equiv_m d$. Unrolling the definitions, we can see that $a - b = km$ and $c - d = jm$ for some $k, j \in \mathbb{Z}$ or equivalently, $a = km + b$ and $c = jm + d$.

Multiplying both together gives us $ac = (km + b)(jm + d) = kjm^2 + kmd + bjm + bd$. 
Modular Arithmetic: Multiplication Property

Let $m$ be a positive integer. If $a \equiv_m b$ and $c \equiv_m d$, then $ac \equiv_m bd$.

Suppose that $a \equiv_m b$ and $c \equiv_m d$. Unrolling the definitions, we can see that $a - b = km$ and $c - d = jm$ for some $k, j \in \mathbb{Z}$ or equivalently, $a = km + b$ and $c = jm + d$.

Multiplying both together gives us $ac = (km + b)(jm + d) = km^2 + kmd + bjm + bd$. Re-arranging, this becomes $ac - bd = m(kjm + kd + bj)$.

This says $ac \equiv_m bd$ by the definition of congruence.
Modular Arithmetic: Properties

If \( a \equiv_m b \) and \( b \equiv_m c \), then \( a \equiv_m c \).

If \( a \equiv_m b \) and \( c \equiv_m d \), then \( a + c \equiv_m b + d \).

Corollary: If \( a \equiv_m b \), then \( a + c \equiv_m b + c \).

If \( a \equiv_m b \) and \( c \equiv_m d \), then \( ac \equiv_m bd \).

Corollary: If \( a \equiv_m b \), then \( ac \equiv_m bc \).
Modular Arithmetic: Properties

If \( a \equiv_m b \) and \( b \equiv_m c \), then \( a \equiv_m c \).

If \( a \equiv_m b \), then \( a + c \equiv_m b + c \).

If \( a \equiv_m b \), then \( ac \equiv_m bc \).

“\( \equiv_m \)” allows us to solve problems in modular arithmetic, e.g.

- add / subtract numbers from both sides of equations
- chains of “\( \equiv_m \)” values shows first and last are “\( \equiv_m \)”
- substitute “\( \equiv_m \)” values in equations (not proven yet)
Properties of “≡ₘ” Used in Algebra

| If $a \equivₘ b$ and $b \equivₘ c$, then $a \equivₘ c$ | “Transitivity” |
| If $a \equivₘ b$ and $c \equivₘ d$, then $a + c \equivₘ b + d$ | “Add Equations” |
| If $a \equivₘ b$ and $c \equivₘ d$, then $ac \equivₘ bd$ | “Multiply Equations” |

These are **Theorems** that we can use in proofs

Example: given that $3x \equivₘ 7$, prove that $5x + 3 \equivₘ 2x + 10$
Properties of “≡_m” Used in Algebra

| If a ≡_m b and b ≡_m c, then a ≡_m c | “Transitivity” |
| If a ≡_m b and c ≡_m d, then a + c ≡_m b + d | “Add Equations” |
| If a ≡_m b and c ≡_m d, then ac ≡_m bd | “Multiply Equations” |

1. 3x ≡_m 7  
   Given
2. 2x = 2x  
   Algebra
3. 2x + 3x ≡_m 2x + 7  
   Add Equations: 2, 1 ??

Line 2 says “=” not “≡_m”

But “=” implies “≡_m”!

(equality is a special case)
Properties of “≡ₘ” Used in Algebra

If \( a \equivₘ b \) and \( b \equivₘ c \), then \( a \equivₘ c \)  “Transitivity”

If \( a \equivₘ b \) and \( c \equivₘ d \), then \( a + c \equivₘ b + d \) “Add Equations”

If \( a \equivₘ b \) and \( c \equivₘ d \), then \( ac \equivₘ bd \) “Multiply Equations”

1. \( 3x \equivₘ 7 \) Given
2. \( 2x = 2x \) Algebra
3. \( 2x \equivₘ 2x \) To Modular: 2
4. \( 2x + 3x \equivₘ 2x + 7 \) Add Equations: 3, 1
5. \( 3 = 3 \) Algebra
6. \( 3 \equivₘ 3 \) To Modular
7. \( 2x + 3x + 3 \equivₘ 2x + 7 + 3 \) Add Equations: 4, 6
Properties of “≡ₘ” Used in Algebra

<table>
<thead>
<tr>
<th>Equation</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. 2x + 3x + 3 ≡ₘ 2x + 7 + 3</td>
<td>Add Equations: 4, 6</td>
</tr>
<tr>
<td>8. 5x + 3 = 2x + 3x + 3</td>
<td>Algebra</td>
</tr>
<tr>
<td>9. 5x + 3 ≡ₘ 2x + 3x + 3</td>
<td>To Modular: 8</td>
</tr>
<tr>
<td>10. 2x + 7 + 3 = 2x + 10</td>
<td>Algebra</td>
</tr>
<tr>
<td>11. 2x + 7 + 3 ≡ₘ 2x + 10</td>
<td>To Modular: 10</td>
</tr>
<tr>
<td>12. 5x + 3 ≡ₘ 2x + 10</td>
<td>Transitivity: 9, 7, 11</td>
</tr>
</tbody>
</table>

Good news! You’ll only have to do this two times in your life...
Properties of “$\equiv_m$” Used in Algebra

<table>
<thead>
<tr>
<th>Condition</th>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $a \equiv_m b$ and $b \equiv_m c$, then $a \equiv_m c$</td>
<td>“Transitivity”</td>
</tr>
<tr>
<td>If $a \equiv_m b$ and $c \equiv_m d$, then $a + c \equiv_m b + d$</td>
<td>“Add Equations”</td>
</tr>
<tr>
<td>If $a \equiv_m b$ and $c \equiv_m d$, then $ac \equiv_m bd$</td>
<td>“Multiply Equations”</td>
</tr>
<tr>
<td>If $a = b$, then $a \equiv_m b$.</td>
<td>“To Modular”</td>
</tr>
</tbody>
</table>

These are **Theorems** that we can use in proofs

**Example:** given that $2(x - 3) \equiv_m 4$, prove that $2x \equiv_m 10$
Properties of “≡_m” Used in Algebra

1. \( 2(x - 3) \equiv_m 4 \)  
   Given

2. \( 2x \equiv_m 10 \)  
   ??
Properties of “$\equiv_m$” Used in Algebra

1. $2(x - 3) \equiv_m 4$
2. $6 = 6$
3. $6 \equiv_m 6$
4. $2(x - 3) + 6 \equiv_m 4 + 6$
5. $2x = 2(x - 3) + 6$
6. $2x \equiv_m 2(x - 3) + 6$
7. $4 + 6 = 10$
8. $4 + 6 \equiv_m 10$
9. $2x \equiv_m 10$

Given
Algebra
To Modular: 2
Add Equations: 1, 3
Algebra
To Modular: 5
Algebra
To Modular: 7
Transitivity: 6, 4, 8
Another Property of “=” Used in Algebra

<table>
<thead>
<tr>
<th>If $a \equiv_m b$ and $b \equiv_m c$, then $a \equiv_m c$</th>
<th>“Transitivity”</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $a \equiv_m b$ and $c \equiv_m d$, then $a + c \equiv_m b + d$</td>
<td>“Add Equations”</td>
</tr>
<tr>
<td>If $a \equiv_m b$ and $c \equiv_m d$, then $ac \equiv_m bd$</td>
<td>“Multiply Equations”</td>
</tr>
<tr>
<td>If $a = b$, then $a \equiv_m b$.</td>
<td>“To Modular”</td>
</tr>
</tbody>
</table>

Can “plug in” (a.k.a. substitute) the known value of a variable

Example: given $2y + 3x = 25$ and $x = 7$, prove that $2y + 21 = 25$. 
Substitution Follows From Other Properties

Given $2y + 3x \equiv_m 25$ and $x \equiv_m 7$, show that $2y + 21 \equiv_m 25$. (substituting 7 for $x$)

Start from $x \equiv_m 7$

Multiply both sides $3x \equiv_m 3 \cdot 7 \quad (= 21)$

Add to both sides $2y + 3x \equiv_m 2y + 21$

Combine $\equiv_m$’s $2y + 21 \equiv_m 2y + 3x \equiv_m 25$
Basic Applications of mod

- Two’s Complement
- Hashing
- Pseudo random number generation
n-bit Unsigned Integer Representation

• Represent integer \( x \) as sum of powers of \( 2 \):

\[
99 = 64 + 32 + 2 + 1 = 2^6 + 2^5 + 2^1 + 2^0 \\
18 = 16 + 2 = 2^4 + 2^1
\]

If \( b_{n-1}2^{n-1} + \cdots + b_12 + b_0 \) with each \( b_i \in \{0,1\} \)
then binary representation is \( b_{n-1} \ldots b_2 b_1 b_0 \)

• For \( n = 8 \):

\[
\begin{align*}
99: & \quad 0110 \ 0011 \\
18: & \quad 0001 \ 0010
\end{align*}
\]

Easy to implement arithmetic \( \text{mod} \ 2^n \)
... just throw away bits \( n+1 \) and up

\[
2^n \mid 2^{n+k} \quad \text{so} \quad b_{n+k}2^{n+k} \equiv 2^n 0 \\
\text{for } k \geq 0
\]
n-bit Unsigned Integer Representation

- Largest representable number is $2^n - 1$

$$2^n = 100\ldots000 \quad \text{(n+1 bits)}$$
$$2^n - 1 = 11\ldots111 \quad \text{(n bits)}$$

32 bits
1 = $0.0001$
$429,496.7295$ max

Berkshire Hathaway’s Stock Price Is Too Much for Computers

Berkshire Hathaway Inc. (BRK-A)
NYSE - Nasdaq Real Time Price. Currency in USD

436,401.00 +679.50 (+0.16%)
Sign-Magnitude Integer Representation

\[ n \text{-bit signed integers} \]

Suppose that \(-2^{n-1} < x < 2^{n-1}\)
First bit as the sign, \(n - 1\) bits for the value

99 = 64 + 32 + 2 + 1
18 = 16 + 2

For \(n = 8\):

\[ \begin{align*}
99: & \quad 0110 0011 \\
-18: & \quad 1001 0010
\end{align*} \]

**Problem**: this has both +0 and -0 (annoying)
Two’s Complement Representation

Suppose that \(0 \leq x < 2^{n-1}\)
\[x\] is represented by the binary representation of \(x\)

Suppose that \(-2^{n-1} \leq x < 0\)
\[x\] is represented by the binary representation of \(x + 2^n\)
result is in the range \(2^{n-1} \leq x < 2^n\)
Two’s Complement Representation

Suppose that $0 \leq x < 2^{n-1}$
   $x$ is represented by the binary representation of $x$
Suppose that $-2^{n-1} \leq x < 0$
   $x$ is represented by the binary representation of $x + 2^n$
result is in the range $2^{n-1} \leq x < 2^n$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>-8</th>
<th>-7</th>
<th>-6</th>
<th>-5</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0001</td>
<td>0010</td>
<td>0011</td>
<td>0100</td>
<td>0101</td>
<td>0110</td>
<td>0111</td>
<td>1000</td>
<td>1001</td>
<td>1010</td>
<td>1011</td>
<td>1100</td>
<td>1101</td>
<td>1110</td>
<td>1111</td>
</tr>
</tbody>
</table>

99 = 64 + 32 + 2 + 1
18 = 16 + 2

For $n = 8$:
   99: 0110 0011
   -18: 1110 1110  (-18 + 256 = 238)
Two’s Complement Representation

Suppose that $0 \leq x < 2^{n-1}$
  $x$ is represented by the binary representation of $x$
Suppose that $-2^{n-1} \leq x < 0$
  $x$ is represented by the binary representation of $x + 2^n$
result is in the range $2^{n-1} \leq x < 2^n$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>-8</th>
<th>-7</th>
<th>-6</th>
<th>-5</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0000</td>
<td>0001</td>
<td>0010</td>
<td>0011</td>
<td>0100</td>
<td>0101</td>
<td>0110</td>
<td>0111</td>
<td>1000</td>
<td>1001</td>
<td>1010</td>
<td>1011</td>
<td>1100</td>
<td>1101</td>
<td>1110</td>
<td>1111</td>
</tr>
</tbody>
</table>

**Key property:** First bit is still the sign bit!

**Key property:** Twos complement representation of any number $y$ is equivalent to $y \mod 2^n$ so arithmetic works $\mod 2^n$

$$y + 2^n \equiv 2^n y$$
Two’s Complement Representation

• For $0 < x \leq 2^{n-1}$, $-x$ is represented by the binary representation of $-x + 2^n$
  
  – How do we calculate $-x$ from $x$?  
  – E.g., what happens for “return $-x$;” in Java?

\[-x + 2^n = (2^n - 1) - x + 1\]

• To compute this, flip the bits of $x$ then add 1!
  
  – All 1’s string is $2^n - 1$, so
    
    Flip the bits of $x$ means replace $x$ by $2^n - 1 - x$
    
    Then add 1 to get $-x + 2^n$
Primes
(and Their Applications)
An integer $p$ greater than 1 is called \textit{prime} if the only positive factors of $p$ are 1 and $p$.

$$p > 1 \land \forall x \left( (x \mid p) \rightarrow ((x = 1) \lor (x = p)) \right)$$

A positive integer that is greater than 1 and is not prime is called \textit{composite}.

$$p > 1 \land \exists x \left( (x \mid p) \land (x \neq 1) \land (x \neq p) \right)$$
Fundamental Theorem of Arithmetic

Every positive integer greater than 1 has a "unique" prime factorization

\[
\begin{align*}
48 &= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 \\
591 &= 3 \cdot 197 \\
45,523 &= 45,523 \\
321,950 &= 2 \cdot 5 \cdot 5 \cdot 47 \cdot 137 \\
1,234,567,890 &= 2 \cdot 3 \cdot 3 \cdot 5 \cdot 3,607 \cdot 3,803
\end{align*}
\]
Algorithmic Problems

• Multiplication
  – Given primes $p_1, p_2, ..., p_k$, calculate their product $p_1 p_2 ... p_k$

• Factoring
  – Given an integer $n$, determine the prime factorization of $n$
Factoring

Factor the following 232 digit number [RSA768]:

123018668453011775513049495838496272077
285356959533479219732245215172640050726
365751874520219978646938995647494277406
384592519255732630345373154826850791702
612214291346167042921431160222124047927
4737794080665351419597459856902143413
Famous Algorithmic Problems

• Factoring  
  – Given an integer $n$, determine the prime factorization of $n$

• Primality Testing  
  – Given an integer $n$, determine if $n$ is prime

• Factoring is hard  
  – (on a classical computer)

• Primality Testing is easy
Greatest Common Divisor
(and Its Applications)
Greatest Common Divisor

GCD(a, b):

Largest integer $d$ such that $d \mid a$ and $d \mid b$

- $\text{GCD}(100, 125) = \quad$ \\
- $\text{GCD}(17, 49) = \quad$ \\
- $\text{GCD}(11, 66) = \quad$ \\
- $\text{GCD}(13, 0) = \quad$ \\
- $\text{GCD}(180, 252) = \quad$

$d$ is GCD iff $(d \mid a) \land (d \mid b) \land \forall x (((x \mid a) \land (x \mid b)) \rightarrow (x \leq d))$
GCD and Factoring

\[ a = 2^3 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11 = 46,200 \]
\[ b = 2 \cdot 3^2 \cdot 5^3 \cdot 7 \cdot 13 = 204,750 \]

\[ \text{GCD}(a, b) = 2^{\min(3,1)} \cdot 3^{\min(1,2)} \cdot 5^{\min(2,3)} \cdot 7^{\min(1,1)} \cdot 11^{\min(1,0)} \cdot 13^{\min(0,1)} \]

Factoring is hard!
Can we compute \textbf{GCD}(a,b) without factoring?
Useful GCD Fact

Let $a$ and $b$ be positive integers. We have $\text{gcd}(a, b) = \text{gcd}(b, a \mod b)$

Proof:
We will show that every number dividing $a$ and $b$ also divides $b$ and $a \mod b$. I.e., $d|a$ and $d|b$ iff $d|b$ and $d|(a \mod b)$.

Hence, their set of common divisors are the same, which means that their greatest common divisor is the same.
Useful GCD Fact

Let $a$ and $b$ be positive integers. We have $\gcd(a, b) = \gcd(b, a \mod b)$

Proof:
By definition of mod, $a = qb + (a \mod b)$ for some integer $q = a \div b$.

Suppose $d|b$ and $d|(a \mod b)$.
Then $b = md$ and $(a \mod b) = nd$ for some integers $m$ and $n$.
Therefore $a = qb + (a \mod b) = qmd + nd = (qm + n)d$.
So $d|a$.

Suppose $d|a$ and $d|b$.
Then $a = kd$ and $b = jd$ for some integers $k$ and $j$.
Therefore $(a \mod b) = a - qb = kd - qjd = (k - qj)d$.
So, $d|(a \mod b)$ also.

Since they have the same common divisors, $\gcd(a, b) = \gcd(b, a \mod b)$. ■
Another simple GCD fact

Let a be a positive integer. We have \( \gcd(a, 0) = a \).
Euclid’s Algorithm

\[ \text{gcd}(a, b) = \text{gcd}(b, a \mod b) \]

\[ \text{gcd}(a, 0) = a \]

```c
int gcd(int a, int b){ /* Assumes: a >= b, b >= 0 */
    if (b == 0) {
        return a;
    } else {
        return gcd(b, a % b);
    }
}
```

Note: \( \text{gcd}(b, a) = \text{gcd}(a, b) \)
Euclid’s Algorithm

Repeatedly use $\text{gcd}(a, b) = \text{gcd}(b, a \mod b)$ to reduce numbers until you get $\text{gcd}(g, 0) = g$.

$\text{gcd}(660, 126) =$
Euclid’s Algorithm

Repeatedly use $\text{gcd}(a, b) = \text{gcd}(b, a \mod b)$ to reduce numbers until you get $\text{gcd}(g, 0) = g$.

$\text{gcd}(660, 126) = \text{gcd}(126, 660 \mod 126) = \text{gcd}(126, 30)$
$= \text{gcd}(30, 126 \mod 30) = \text{gcd}(30, 6)$
$= \text{gcd}(6, 30 \mod 6) = \text{gcd}(6, 0)$
$= 6$
Bézout’s theorem

If $a$ and $b$ are positive integers, then there exist integers $s$ and $t$ such that

$$\text{gcd}(a,b) = sa + tb.$$
Extended Euclidean algorithm

- Can use Euclid’s Algorithm to find $s, t$ such that

$$\gcd(a, b) = sa + tb$$
Extended Euclidean algorithm

• Can use Euclid's Algorithm to find $s, t$ such that
  \[ \gcd(a, b) = sa + tb \]

Step 1 (Compute GCD & Keep Tableau Information):

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>b</th>
<th>a mod b = r</th>
<th>b</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>27</td>
<td>27</td>
<td>35 mod 27</td>
<td>27</td>
<td>8</td>
</tr>
</tbody>
</table>

\[ \gcd(35, 27) = \gcd(27, 35 \mod 27) = \gcd(27, 8) \]

\[ 35 = 1 \times 27 + 8 \]
Extended Euclidean algorithm

• Can use Euclid’s Algorithm to find $s, t$ such that

$$\text{gcd}(a, b) = sa + tb$$

**Step 1 (Compute GCD & Keep Tableau Information):**

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>b</th>
<th>a mod b</th>
<th>b = r</th>
<th>b</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>27</td>
<td>27</td>
<td>35 mod 27</td>
<td>8</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27</td>
<td>27 mod 8</td>
<td>8</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>3 mod 8</td>
<td>3</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>2 mod 3</td>
<td>2</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1 mod 2</td>
<td>1</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0 mod 1</td>
<td>1</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

$35 = 1 \times 27 + 8$
$27 = 3 \times 8 + 3$
$8 = 2 \times 3 + 2$
$3 = 1 \times 2 + 1$
Extended Euclidean algorithm

- Can use Euclid’s Algorithm to find $s, t$ such that
  \[ \text{gcd}(a, b) = sa + tb \]

Step 2 (Solve the equations for $r$):

\[
\begin{align*}
\text{a} & = q \times \text{b} + r \\
35 & = 1 \times 27 + 8 \\
27 & = 3 \times 8 + 3 \\
8 & = 2 \times 3 + 2 \\
3 & = 1 \times 2 + 1
\end{align*}
\]
Extended Euclidean algorithm

- Can use Euclid’s Algorithm to find $s, t$ such that

\[ \gcd(a, b) = sa + tb \]

Step 2 (Solve the equations for $r$):

\[
\begin{align*}
  a &= q \times b + r \\
  35 &= 1 \times 27 + 8 \\
  27 &= 3 \times 8 + 3 \\
  8 &= 2 \times 3 + 2 \\
  3 &= 1 \times 2 + 1
\end{align*}
\]

\[
\begin{align*}
  r &= a - q \times b \\
  8 &= 35 - 1 \times 27 \\
  3 &= 27 - 3 \times 8 \\
  2 &= 8 - 2 \times 3 \\
  1 &= 3 - 1 \times 2
\end{align*}
\]
Extended Euclidean algorithm

- Can use Euclid’s Algorithm to find $s, t$ such that
  \[ \gcd(a, b) = sa + tb \]

**Step 3 (Backward Substitute Equations):**

\[
\begin{align*}
8 &= 35 - 1 \times 27 \\
3 &= 27 - 3 \times 8 \\
2 &= 8 - 2 \times 3 \\
\boxed{1} &= 3 - 1 \times 2
\end{align*}
\]
Extended Euclidean algorithm

• Can use Euclid’s Algorithm to find $s, t$ such that

$$\gcd(a, b) = sa + tb$$

Step 3 (Backward Substitute Equations):

$$8 = 35 - 1 \times 27$$

$$3 = 27 - 3 \times 8$$

$$2 = 8 - 2 \times 3$$

$$1 = 3 - 1 \times 2$$

1. Plug in the def of 2
2. Re-arrange into 3’s and 8’s
Extended Euclidean algorithm

- Can use Euclid’s Algorithm to find $s, t$ such that
  
  \[ \gcd(a, b) = sa + tb \]

**Step 3 (Backward Substitute Equations):**

\[
8 = 35 - 1 \times 27 \\
3 = 27 - 3 \times 8 \\
2 = 8 - 2 \times 3 \\
1 = 3 - 1 \times 2
\]

Plug in the def of 2

Re-arrange into 3’s and 8’s

Plug in the def of 3

Re-arrange into 8’s and 27’s
Extended Euclidean algorithm

- Can use Euclid’s Algorithm to find \( s, t \) such that
  \[
gcd(a, b) = sa + tb
\]

**Step 3 (Backward Substitute Equations):**

1. \( 8 = 35 - 1 \times 27 \)
2. \( 3 = 27 - 3 \times 8 \)
3. \( 2 = 8 - 2 \times 3 \)
4. \( 1 = 3 - 1 \times 2 \)

- Re-arrange into 3's and 8's
- Re-arrange into 27's and 35's
- Plug in the def of 3
- Plug in the def of 2

\[
1 = 3 - 1 \times (8 - 2 \times 3) = 3 - 8 + 2 \times 3 = (-1) \times 8 + 3 \times 3
\]

\[
2 = 8 - 2 \times 3 = (-1) \times 8 + 3 \times (27 - 3 \times 8)
\]

\[
3 = 27 - 3 \times 8 = (-1) \times 8 + 3 \times 27 + (-9) \times 8
\]

\[
8 = 35 - 1 \times 27 = 3 \times 27 + (-10) \times 8
\]

\[
1 = 3 \times 27 + (-10) \times (35 - 1 \times 27) = 13 \times 27 + (-10) \times 35
\]
Multiplicative inverse mod m

Let $0 \leq a, b < m$. Then, $b$ is the multiplicative inverse of $a$ (modulo $m$) iff $ab \equiv_m 1$. 

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

mod 7

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

mod 10
**Multiplicative inverse mod** \( m \)

Suppose \( \gcd(a, m) = 1 \)

By Bézout’s Theorem, there exist integers \( s \) and \( t \) such that \( sa + tm = 1 \).

\( s \) is the multiplicative inverse of \( a \) (modulo \( m \)): \[ 1 = sa + tm \equiv_m sa \]

So... we can compute multiplicative inverses with the extended Euclidean algorithm

These inverses let us solve modular equations...
Example: Solve a Modular Equation

Solve: $7x \equiv_{26} 3$  
Find multiplicative inverse of 7 modulo 26
Example: Solve a Modular Equation

Solve: $7x \equiv_{26} 3$  
Find multiplicative inverse of 7 modulo 26

$$\text{gcd}(26, 7) = \text{gcd}(7, 5) = \text{gcd}(5, 2) = \text{gcd}(2, 1) = 1$$
Example: Solve a Modular Equation

Solve: $7x \equiv_{26} 3$    \hspace{1cm} \text{Find multiplicative inverse of } 7 \text{ modulo } 26

\[
\begin{align*}
gcd(26, 7) &= gcd(7, 5) = gcd(5, 2) = gcd(2, 1) = 1 \\
26 &= 3 \times 7 + 5 \\
7 &= 1 \times 5 + 2 \\
5 &= 2 \times 2 + 1
\end{align*}
\]
Example: Solve a Modular Equation

Solve: \(7x \equiv_{26} 3\) \hspace{1cm} \text{Find multiplicative inverse of 7 modulo 26}

\[
gcd(26, 7) = gcd(7, 5) = gcd(5, 2) = gcd(2, 1) = 1
\]

\[
26 = 3 \times 7 + 5 \quad 5 = 26 - 3 \times 7
\]

\[
7 = 1 \times 5 + 2 \quad 2 = 7 - 1 \times 5
\]

\[
5 = 2 \times 2 + 1 \quad 1 = 5 - 2 \times 2
\]
Example: Solve a Modular Equation

Solve: \( 7x \equiv_{26} 3 \) Find multiplicative inverse of 7 modulo 26

\[
gcd(26, 7) = gcd(7, 5) = gcd(5, 2) = gcd(2, 1) = 1
\]

\[
26 = 3 \times 7 + 5 \\
5 = 26 - 3 \times 7
\]

\[
7 = 1 \times 5 + 2 \\
2 = 7 - 1 \times 5
\]

\[
5 = 2 \times 2 + 1 \\
1 = 5 - 2 \times 2
\]

\[
1 = 5 - 2 \times (7 - 1 \times 5)
\]

\[
= (-2) \times 7 + 3 \times 5
\]

\[
= (-2) \times 7 + 3 \times (26 - 3 \times 7)
\]

\[
= (-11) \times 7 + 3 \times 26
\]
Example: Solve a Modular Equation

Solve: \( 7x \equiv_{26} 3 \)

Find multiplicative inverse of 7 modulo 26

\[
\text{gcd}(26, 7) = \text{gcd}(7, 5) = \text{gcd}(5, 2) = \text{gcd}(2, 1) = 1
\]

\[
26 = 3 \times 7 + 5 \quad 5 = 26 - 3 \times 7
\]

\[
7 = 1 \times 5 + 2 \quad 2 = 7 - 1 \times 5
\]

\[
5 = 2 \times 2 + 1 \quad 1 = 5 - 2 \times 2
\]

\[
1 = 5 - 2 \times (7 - 1 \times 5)
\]

\[
= (-2) \times 7 + 3 \times 5
\]

\[
= (-2) \times 7 + 3 \times (26 - 3 \times 7)
\]

\[
= (-11) \times 7 + 3 \times 26
\]

“the” multiplicative inverse

Now \((-11) \mod 26 = 15.\)

\((-11\) is also “a” multiplicative inverse)
Example: Solve a Modular Equation

Solve: \( 7x \equiv_{26} 3 \)

Find multiplicative inverse of 7 modulo 26... it’s 15.

Multiplying both sides by 15 gives

\[
15 \cdot 7x \equiv_{26} 15 \cdot 3
\]

Simplify on both sides to get

\[
x \equiv_{26} 15 \cdot 7x \equiv_{26} 15 \cdot 3 \equiv_{26} 19
\]

So, all solutions of this congruence are numbers of the form \( x = 19 + 26k \) for some \( k \in \mathbb{Z} \).
Example: Solve a Modular Equation

Solve: \( 7x \equiv_{26} 3 \)

Conversely, suppose that \( x \equiv_{26} 19 \).

Multiplying both sides by 7 gives

\[
7x \equiv_{26} 7 \cdot 19
\]

Simplify on right to get

\[
7x \equiv_{26} 7 \cdot 19 \equiv_{26} 3
\]

So, all numbers of form \( x = 19 + 26k \) for any \( k \in \mathbb{Z} \) are solutions of this equation.
Example: Solve a Modular Equation

Solve: \( 7x \equiv_{26} 3 \) (on HW or exams)

Step 1. Find multiplicative inverse of 7 modulo 26

\[
1 = \ldots = (-11) \cdot 7 + 3 \cdot 26
\]

Since \((-11) \mod 26 = 15\), the inverse of 7 is 15.

Step 2. Multiply both sides and simplify

Multiplying by 15, we get \( x \equiv_{26} 15 \cdot 7x \equiv_{26} 15 \cdot 3 \equiv_{26} 19 \).

Step 3. State the full set of solutions

So, the solutions are \( 19 + 26k \) for any \( k \in \mathbb{Z} \)

(must be of the form \( a + mk \) for all \( k \in \mathbb{Z} \) with \( 0 \leq a < m \))
Examples Not in “Standard Form”

Solve: \( 7x \equiv_{26} 3 \)

Modular equation like \( Ax \equiv_{26} B \) for some \( A \) and \( B \) is in “standard form”.

– solve by multiplying both sides by inverse of \( A \)

What about other equations like

\[
7(x - 3) \equiv_{26} 8
\]

Previously saw how to formally prove this has the same solutions as equation above.
Examples Not in “Standard Form”

Solve: \[ 7x \equiv_{26} 3 \]

Modular equation like \[ Ax \equiv_{26} B \] for some \( A \) and \( B \) is in “standard form”.

- solve by multiplying both sides by inverse of \( A \)

What about other equations like

\[ 7(x - 3) \equiv_{26} 8 \]?

On HW4:

- apply algorithm when in standard form (English)
- transform non-standard to standard form (formal)
Math mod a prime is especially nice

gcd(a, m) = 1 if m is prime and 0 < a < m so can always solve these equations mod a prime.

\[
\begin{array}{cccccccc}
+ & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1 & 2 & 3 & 4 & 5 & 6 & 0 \\
2 & 2 & 3 & 4 & 5 & 6 & 0 & 1 \\
3 & 3 & 4 & 5 & 6 & 0 & 1 & 2 \\
4 & 4 & 5 & 6 & 0 & 1 & 2 & 3 \\
5 & 5 & 6 & 0 & 1 & 2 & 3 & 4 \\
6 & 6 & 0 & 1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
x & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
2 & 0 & 2 & 4 & 6 & 1 & 3 & 5 \\
3 & 0 & 3 & 6 & 2 & 5 & 1 & 4 \\
4 & 0 & 4 & 1 & 5 & 2 & 6 & 3 \\
5 & 0 & 5 & 3 & 1 & 6 & 4 & 2 \\
6 & 0 & 6 & 5 & 4 & 3 & 2 & 1 \\
\end{array}
\]

mod 7
Adding to both sides is an equivalence:

\[ x \equiv_m y \]

\[ x + c \equiv_m y + c \]

The same is not true of multiplication...

unless we have a multiplicative inverse \( cd \equiv_m 1 \)

\[ cx \equiv_m cy \]
Modular Exponentiation mod 7

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>a^1</th>
<th>a^2</th>
<th>a^3</th>
<th>a^4</th>
<th>a^5</th>
<th>a^6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
Exponentiation

• Compute $78365^{81453}$

• Compute $78365^{81453} \mod 104729$

• Output is small
  – need to keep intermediate results small
Small Multiplications

Since $b = qm + (b \mod m)$, we have $b \mod m \equiv_m b$.

And since $c = tm + (c \mod m)$, we have $c \mod m \equiv_m c$.

Multiplying these gives $(b \mod m)(c \mod m) \equiv_m bc$.

By the Lemma from a few lectures ago, this tells us $bc \mod m = (b \mod m)(c \mod m) \mod m$.

Okay to mod $b$ and $c$ by $m$ before multiplying if we are planning to mod the result by $m$. 
Repeated Squaring – small and fast

Since \( b \mod m \equiv_m b \) and \( c \mod m \equiv_m c \)
we have \( bc \mod m = (b \mod m)(c \mod m) \mod m \)

So \( a^2 \mod m = (a \mod m)^2 \mod m \)
and \( a^4 \mod m = (a^2 \mod m)^2 \mod m \)
and \( a^8 \mod m = (a^4 \mod m)^2 \mod m \)
and \( a^{16} \mod m = (a^8 \mod m)^2 \mod m \)
and \( a^{32} \mod m = (a^{16} \mod m)^2 \mod m \)

Can compute \( a^k \mod m \) for \( k = 2^i \) in only \( i \) steps

What if \( k \) is not a power of \( 2 \)?
Fast Exponentiation Algorithm

81453 in binary is 10011111000101101

\[ 81453 = 2^{16} + 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^9 + 2^5 + 2^3 + 2^2 + 2^0 \]

\[ a^{81453} = a^{2^{16}} \cdot a^{2^{13}} \cdot a^{2^{12}} \cdot a^{2^{11}} \cdot a^{2^{10}} \cdot a^9 \cdot a^5 \cdot a^3 \cdot a^2 \cdot a^0 \]

\[ a^{81453} \mod m = (\ldots( (((a^{2^{16}} \mod m \cdot a^{2^{13}} \mod m) \mod m \cdot a^{2^{12}} \mod m) \mod m \cdot a^{2^{11}} \mod m) \mod m \cdot a^{2^{10}} \mod m) \mod m \cdot a^9 \mod m) \mod m \cdot a^5 \mod m) \mod m \cdot a^3 \mod m) \mod m \cdot a^2 \mod m) \mod m \cdot a^0 \mod m) \mod m \]

Uses only 16 + 9 = 25 multiplications

The fast exponentiation algorithm computes

\[ a^k \mod m \text{ using } \leq 2 \log k \text{ multiplications mod } m \]
Fast Exponentiation: $a^k \mod m$ for all $k$

Another way....

$$a^{2j} \mod m = (a^j \mod m)^2 \mod m$$

$$a^{2j+1} \mod m = ((a \mod m) \cdot (a^{2j} \mod m)) \mod m$$
Fast Exponentiation

public static int FastModExp(int a, int k, int modulus) {
    if (k == 0) {
        return 1;
    } else if ((k % 2) == 0) {
        long temp = FastModExp(a, k/2, modulus);
        return (temp * temp) % modulus;
    } else {
        long temp = FastModExp(a, k-1, modulus);
        return (a * temp) % modulus;
    }
}

\[
\begin{align*}
    a^{2j} \mod m &= (a^j \mod m)^2 \mod m \\
    a^{2j+1} \mod m &= ((a \mod m) \cdot (a^{2j} \mod m)) \mod m
\end{align*}
\]
Using Fast Modular Exponentiation

- Your e-commerce web transactions use SSL (Secure Socket Layer) based on RSA encryption
- RSA
  - Vendor chooses random 512-bit or 1024-bit primes $p, q$ and 512/1024-bit exponent $e$. Computes $m = p \cdot q$
  - Vendor broadcasts $(m, e)$
  - To send $a$ to vendor, you compute $C = a^e \mod m$ using fast modular exponentiation and send $C$ to the vendor.
  - Using secret $p, q$ the vendor computes $d$ that is the multiplicative inverse of $e \mod (p - 1)(q - 1)$.
  - Vendor computes $C^d \mod m$ using fast modular exponentiation.
- Fact: $a = C^d \mod m$ for $0 < a < m$ unless $p | a$ or $q | a$