Lecture #11: Integers and Modular Arithmetic

Based on materials developed by Dr. Adam Lee
Today’s Topics

Integers and division

- The division algorithm
- Modular arithmetic
- Applications of modular arithmetic
Number theory is the branch of mathematics that explores the integers and their properties.

Number theory has many applications within computer science, including:

- Organizing data
- Encrypting sensitive data
- Developing error correcting codes
- Generating “random” numbers
- ...  

We will only scratch the surface...
The notion of divisibility is one of the most basic properties of the integers

**Definition:** If $a$ and $b$ are integers and $a \neq 0$, we say that $a$ divides $b$ if there is an integer $c$ such that $b = ac$. We write $a \mid b$ to say that $a$ divides $b$, and $a \nmid b$ to say that $a$ does not divide $b$.

**Mathematically:** $a \mid b \iff \exists c \in \mathbb{Z} \ (b = ac)$

**Note:** If $a \mid b$, then

- $a$ is called a **factor** of $b$
- $b$ is called a **multiple** of $a$

We’ve been using the notion of divisibility all along!

- $E = \{x \mid x = 2k \land k \in \mathbb{Z}\}$
Division examples

Examples:
- Does 4 | 16?
- Does 3 | 11?
- Does 7 | 42?

Question: Let $n$ and $d$ be two positive integers. How many positive integers not exceeding $n$ are divisible by $d$?

Answer: We want to count the number of integers of the form $dk$ that are less than $n$. That is, we want to know the number of integers $k$ with $0 \leq dk \leq n$, or $0 \leq k \leq n/d$. Therefore, there are $\lfloor n/d \rfloor$ positive integers not exceeding $n$ that are divisible by $d$. 
Important properties of divisibility

Property 1: If \( a \mid b \) and \( a \mid c \), then \( a \mid (b + c) \)

Property 2: If \( a \mid b \), then \( a \mid bc \) for all integers \( c \).

Property 3: If \( a \mid b \) and \( b \mid c \), then \( a \mid c \).
**Theorem:** Let $a$ be an integer and let $d$ be a positive integer. There are unique integers $q$ and $r$, with $0 \leq r < d$, such that $a = dq + r$.

For historical reasons, the above theorem is called the **division algorithm**, even though it isn’t an algorithm!

**Terminology:** Given $a = dq + r$

- $a$ is called the **dividend**
- $d$ is called the **divisor**
- $q$ is called the **quotient**
- $r$ is called the **remainder**
- $q = a \text{ div } d$
- $r = a \text{ mod } d$

*div and mod are operators*
Examples

**Question:** What are the quotient and remainder when 123 is divided by 23?

**Answer:** We have that $123 = 23 \times 5 + 8$. So the quotient is $123 \div 23 = 5$, and the remainder is $123 \mod 23 = 8$.

---

**Question:** What are the quotient and remainder when -11 is divided by 3?

**Answer:** Since $-11 = 3 \times -4 + 1$, we have that the quotient is -4 and the remainder is 1.

Recall that since the remainder must be non-negative, $3 \times -3 - 2$ is not a valid use of the division theorem!
In-class exercises

Problems 1-4: On Top Hat
Sometimes, we care only about the remainder of an integer after it is divided by some other integer.

**Example:** What time will it be 22 hours from now?

**Answer:** If it is 6am now, it will be \((6 + 22) \mod 24 = 28 \mod 24 = 4\) am in 22 hours.
Since remainders can be so important, they have their own special notation!

**Definition:** If $a$ and $b$ are integers and $m$ is a positive integer, we say that $a$ is congruent to $b$ modulo $m$ if $m \mid (a - b)$. We write this as $a \equiv b \pmod{m}$.

**Note:** $a \equiv b \pmod{m}$ iff $a \mod m = b \mod m$.

**Examples:**
- Is 17 congruent to 5 modulo 6?
- Is 24 congruent to 14 modulo 6?
Properties of congruencies

**Theorem:** Let $m$ be a positive integer. The integers $a$ and $b$ are congruent modulo $m$ ($a \equiv b \pmod{m}$) iff there is an integer $k$ such that $a = b + km$.

**Theorem:** Let $m$ be a positive integer. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then

- $(a + c) \equiv (b + d) \pmod{m}$
- $ac \equiv bd \pmod{m}$
Congruencies have many applications within computer science

Today we’ll look at three:

1. Hash functions
2. The generation of pseudorandom numbers
3. Cryptography
Hash functions allow us to quickly and efficiently locate data

**Problem:** Given a large collection of records, how can we find the one we want quickly?

**Solution:** Apply a hash function that determines the storage location of the record based on the record’s ID. A common hash function is $h(k) = k \mod n$, where $n$ is the number of available storage locations.

<table>
<thead>
<tr>
<th>Memory:</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>
</tr>
</thead>
<tbody>
<tr>
<td>ID: 42</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID: 276</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID: 23</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$42 \mod 8 = 2$  
$276 \mod 8 = 4$  
$23 \mod 8 = 7$
Many areas of computer science rely on the ability to generate pseudorandom numbers.

- Hardware, software, and network simulation
- Coding algorithms
- Security
- Network protocols
Congruencies can be used to generate pseudorandom sequences

**Step 1:** Choose
- A modulus $m$
- A multiplier $a$
- An increment $c$
- A seed $x_0$

**Step 2:** Apply the following
- $x_{n+1} = (ax_n + c) \mod m$

**Example:** $m = 9$, $a = 7$, $c = 4$, $x_0 = 3$
- $x_1 = 7x_0 + 4 \mod 9 = 7 \times 3 + 4 \mod 9 = 25 \mod 9 = 7$
- $x_2 = 7x_1 + 4 \mod 9 = 7 \times 7 + 4 \mod 9 = 53 \mod 9 = 8$
- $x_3 = 7x_2 + 4 \mod 9 = 7 \times 8 + 4 \mod 9 = 60 \mod 9 = 6$
- $x_4 = 7x_3 + 4 \mod 9 = 7 \times 6 + 4 \mod 9 = 46 \mod 9 = 1$
- $x_5 = 7x_4 + 4 \mod 9 = 7 \times 1 + 4 \mod 9 = 11 \mod 9 = 2$
- ...

The field of cryptography makes heavy use of number theory and congruencies

Cryptography is the study of secret messages

Uses of cryptography:

- Protecting medical records
- Storing and transmitting military secrets
- Secure web browsing
- ...

Congruencies are used in cryptosystems from antiquity, as well as in modern-day algorithms

Since modern algorithms require quite a bit of background to discuss, we’ll examine an ancient cryptosystem
The Caesar cipher is based on congruencies.

To encode a message using the Caesar cipher:

- Choose a shift index $s$
- Convert each letter A-Z into a number 0-25
- Compute $f(p) = p + s \mod 26$

**Example:** Let $s = 9$. Encode “ATTACK”.

- ATTACK = 0 19 19 0 2 10
- $f(0) = 9$, $f(19) = 2$, $f(2) = 11$, $f(10) = 19$
- Encrypted message: 9 2 2 9 11 19 = JCCJLT
Decryption involves using the inverse function

That is, \( f^{-1}(p) = p - s \mod 26 \)

**Example:** Assume that \( s = 3 \). Decrypt the message “UHWUHDW”.

- UHWUHDW = 20 7 22 20 7 3 22
- \( f^{-1}(20) = 17 \), \( f^{-1}(7) = 4 \), \( f^{-1}(22) = 19 \), \( f^{-1}(3) = 0 \)
- Decrypted result: 17 4 19 17 4 0 19 = RETREAT
In-class exercises

Problem 5-7: On Top Hat
Number theory is the study of integers and their properties

Divisibility, modular arithmetic, and congruency are used throughout computer science

Next time:
- Prime numbers, GCDs (Section 4.3)