



CSE 311 Lecture 11: Modular Arithmetic

Emina Torlak and Kevin Zatloukal

Topics

Sets and set operations

A quick wrap-up of [Lecture 10](#).

Modular arithmetic basics

Arithmetic over a finite domain (a.k.a computer arithmetic).

Modular arithmetic properties

Congruence, addition, multiplication, proofs.

Modular arithmetic and integer representations

Unsigned, sign-magnitude, and two's complement representation.

Applications of modular arithmetic

Hashing, pseudo-random numbers, ciphers.

Sets and set operations

A quick wrap-up of [Lecture 10](#).

Sets and set operations

A set is a collection of *elements*.

Write $a \in B$ (or $a \notin B$) to say that a is (or isn't) an element in the set B .

The order of elements doesn't matter, and duplicates don't matter.

Sets **A** and **B** are *equal* if they have the same elements.

$$A = B \equiv \forall x. x \in A \leftrightarrow x \in B$$

A is a *subset* of **B** if every element of **A** is also in **B**.

$$A \subseteq B \equiv \forall x. x \in A \rightarrow x \in B$$

Sets can be built from predicates or using set operations.

$$A \cup B = \{x : (x \in A) \vee (x \in B)\}$$

$$A \cap B = \{x : (x \in A) \wedge (x \in B)\}$$

$$A \setminus B = \{x : (x \in A) \wedge (x \notin B)\}$$

$$A \oplus B = \{x : (x \in A) \oplus (x \in B)\}$$

$$\overline{A} = \{x : x \notin A\} = \{x : \neg(x \in A)\}$$

This is Boolean algebra again

Union \cup is defined using \vee .

$$A \cup B = \{x : (x \in A) \vee (x \in B)\}$$

Intersection \cap is defined using \wedge .

$$A \cap B = \{x : (x \in A) \wedge (x \in B)\}$$

Complement works like \neg .

$$\bar{A} = \{x : x \notin A\} = \{x : \neg(x \in A)\}$$

This means that all equivalences from Boolean algebra translate directly into set theory, and you can use them in your proofs!

More sets

Power set, Cartesian product, and Russell's paradox.

Power set

Power set of a set A is the set of all subsets of A .

$$\mathcal{P}(A) = \{B : B \subseteq A\}$$

Examples

Let $\text{Days} = \{M, W, F\}$.

$$\mathcal{P}(\text{Days}) =$$

$$\mathcal{P}(\emptyset) =$$

Power set

Power set of a set A is the set of all subsets of A .

$$\mathcal{P}(A) = \{B : B \subseteq A\}$$

Examples

Let Days = $\{M, W, F\}$.

$$\mathcal{P}(\text{Days}) = \{\emptyset, \{M\}, \{W\}, \{F\}, \{M, W\}, \{M, F\}, \{W, F\}, \{M, W, F\}\}$$

$$\mathcal{P}(\emptyset) =$$

Power set

Power set of a set A is the set of all subsets of A .

$$\mathcal{P}(A) = \{B : B \subseteq A\}$$

Examples

Let Days = $\{M, W, F\}$.

$$\mathcal{P}(\text{Days}) = \{\emptyset, \{M\}, \{W\}, \{F\}, \{M, W\}, \{M, F\}, \{W, F\}, \{M, W, F\}\}$$

$$\mathcal{P}(\emptyset) = \{\emptyset\} \neq \emptyset$$

Cartesian product

The Cartesian product of two sets is the set of all of their ordered pairs.

$$A \times B = \{(a, b) : a \in A \wedge b \in B\}$$

Examples

$\mathbb{R} \times \mathbb{R}$ is the real plane.

$\mathbb{Z} \times \mathbb{Z}$ is the set of all pairs of integers.

Cartesian product

The Cartesian product of two sets is the set of all of their ordered pairs.

$$A \times B = \{(a, b) : a \in A \wedge b \in B\}$$

Examples

$\mathbb{R} \times \mathbb{R}$ is the real plane.

$\mathbb{Z} \times \mathbb{Z}$ is the set of all pairs of integers.

If $A = \{1, 2\}$, $B = \{a, b, c\}$,

then $A \times B =$

Cartesian product

The Cartesian product of two sets is the set of all of their ordered pairs.

$$A \times B = \{(a, b) : a \in A \wedge b \in B\}$$

Examples

$\mathbb{R} \times \mathbb{R}$ is the real plane.

$\mathbb{Z} \times \mathbb{Z}$ is the set of all pairs of integers.

If $A = \{1, 2\}$, $B = \{a, b, c\}$,

then $A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$.

Cartesian product

The Cartesian product of two sets is the set of all of their ordered pairs.

$$A \times B = \{(a, b) : a \in A \wedge b \in B\}$$

Examples

$\mathbb{R} \times \mathbb{R}$ is the real plane.

$\mathbb{Z} \times \mathbb{Z}$ is the set of all pairs of integers.

If $A = \{1, 2\}$, $B = \{a, b, c\}$,

then $A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$.

$A \times \emptyset =$

Cartesian product

The Cartesian product of two sets is the set of all of their ordered pairs.

$$A \times B = \{(a, b) : a \in A \wedge b \in B\}$$

Examples

$\mathbb{R} \times \mathbb{R}$ is the real plane.

$\mathbb{Z} \times \mathbb{Z}$ is the set of all pairs of integers.

If $A = \{1, 2\}$, $B = \{a, b, c\}$,

then $A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$.

$A \times \emptyset = \{(a, b) : a \in A \wedge b \in \emptyset\} = \{(a, b) : a \in A \wedge \text{F}\} = \emptyset$.

Russell's paradox

Let S be the set of all sets that don't contain themselves.

$$S = \{x : x \notin x\}$$

Russell's paradox

Let S be the set of all sets that don't contain themselves.

$$S = \{x : x \notin x\}$$

The definition of S is contradictory, hence the paradox.

Russell's paradox

Let S be the set of all sets that don't contain themselves.

$$S = \{x : x \notin x\}$$

The definition of S is contradictory, hence the paradox.

Suppose that $S \in S$. Then, by definition of S , $S \notin S$, which is a contradiction.

Russell's paradox

Let S be the set of all sets that don't contain themselves.

$$S = \{x : x \notin x\}$$

The definition of S is contradictory, hence the paradox.

Suppose that $S \in S$. Then, by definition of S , $S \notin S$, which is a contradiction.

Suppose that $S \notin S$. Then, by definition of S , $S \in S$, which is a contradiction too.

Russell's paradox

Let S be the set of all sets that don't contain themselves.

$$S = \{x : x \notin x\}$$

The definition of S is contradictory, hence the paradox.

Suppose that $S \in S$. Then, by definition of S , $S \notin S$, which is a contradiction.

Suppose that $S \notin S$. Then, by definition of S , $S \in S$, which is a contradiction too.

To avoid the paradox ...

Define S with respect to a universe of discourse.

$$S = \{x \in U : x \notin x\}$$

With this definition, $S \notin S$ and there is no contradiction because $S \notin U$.

Working with sets

Representing sets as bitvectors and applications of bitvectors.

Representing sets as bitvectors

Suppose that universe U is $\{1, 2, \dots, n\}$.

We can represent every set $B \subseteq U$ as a vector of bits:

$$b_1 b_2 \dots b_n \text{ where } b_i = 1 \text{ if } i \in B$$
$$b_i = 0 \text{ if } i \notin B$$

This is called the *characteristic vector* of set B .

Representing sets as bitvectors

Suppose that universe U is $\{1, 2, \dots, n\}$.

We can represent every set $B \subseteq U$ as a vector of bits:

$$b_1 b_2 \dots b_n \text{ where } b_i = 1 \text{ if } i \in B$$
$$b_i = 0 \text{ if } i \notin B$$

This is called the *characteristic vector* of set B .

Given characteristic vectors for A and B , what is the vector for

$$A \cup B =$$

$$A \cap B =$$

Representing sets as bitvectors

Suppose that universe U is $\{1, 2, \dots, n\}$.

We can represent every set $B \subseteq U$ as a vector of bits:

$$b_1 b_2 \dots b_n \text{ where } b_i = 1 \text{ if } i \in B$$
$$b_i = 0 \text{ if } i \notin B$$

This is called the *characteristic vector* of set B .

Given characteristic vectors for A and B , what is the vector for

$$A \cup B = (a_1 + b_1) \dots (a_n + b_n)$$

$$A \cap B =$$

Representing sets as bitvectors

Suppose that universe U is $\{1, 2, \dots, n\}$.

We can represent every set $B \subseteq U$ as a vector of bits:

$$b_1 b_2 \dots b_n \text{ where } b_i = 1 \text{ if } i \in B$$
$$b_i = 0 \text{ if } i \notin B$$

This is called the *characteristic vector* of set B .

Given characteristic vectors for \mathbf{A} and \mathbf{B} , what is the vector for

$$A \cup B = (a_1 + b_1) \dots (a_n + b_n)$$

$$A \cap B = (a_1 \cdot b_1) \dots (a_n \cdot b_n)$$

Unix/Linux file permissions

```
$ ls -l  
drwxr-xr-x ... Documents/  
-rw-r--r-- ... file1
```

Permissions maintained as bitvectors.

Letter means the bit is 1.

”-“ means the bit is zero.

Bitwise operations

$$\begin{array}{r} 01101101 \\ \vee 00110111 \\ \hline 01111111 \end{array}$$

$$z = x \mid y$$

$$\begin{array}{r} 00101010 \\ \wedge 00001111 \\ \hline 00001010 \end{array}$$

$$z = x \& y$$

$$\begin{array}{r} 01101101 \\ \oplus 00110111 \\ \hline 01011010 \end{array}$$

$$z = x \hat{=} y$$

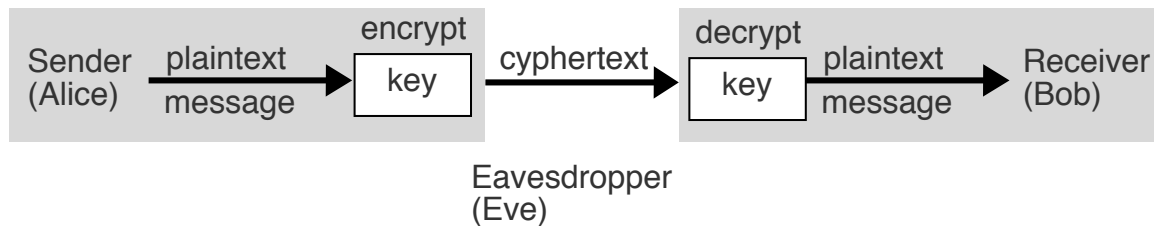
Note that $(x \oplus y) \oplus y = x$.

Private key cryptography

Alice wants to communicate a message m secretly to Bob, so that eavesdropper Eve who sees their conversation can't understand m .

Alice and Bob can get together ahead of time and privately share a secret key K .

How can they communicate securely in this setting?



One-time pad

Alice and Bob privately share a random n -bitvector \mathbf{K} .

Eve doesn't know K .

Later, Alice has n -bit message \mathbf{m} to send to Bob.

Alice computes $C = m \oplus K$.

Alice sends C to Bob.

Bob computes $m = C \oplus K$, which is $(m \oplus K) \oplus K = m$.

Eve can't figure out \mathbf{m} from \mathbf{C} unless she can guess \mathbf{K} .

And that's very unlikely for large n ...

Modular arithmetic basics

Arithmetic over a finite domain (a.k.a computer arithmetic).

Modular arithmetic in action

What does [this Java program](#) output?

```
public class Seconds {
    final static int SEC_IN_YEAR = 364*24*60*60;
    public static void main(String args[]) {
        System.out.println("Number of seconds in 10100 years: " +
            (SEC_IN_YEAR*10100));
    }
}
```

Modular arithmetic in action

What does [this Java program](#) output?

```
public class Seconds {
    final static int SEC_IN_YEAR = 364*24*60*60;
    public static void main(String args[]) {
        System.out.println("Number of seconds in 10100 years: " +
            (SEC_IN_YEAR*10100));
    }
}
```

```
$javac Test.java
$java -Xmx128M -Xms16M Test
Number of seconds in 10100 years: -186619904
```

You'll recognize this as “integer overflow.” It happens because computers use *modular arithmetic* to operate on finite integer data types, such as `int`.

It all starts with divisibility ...

Divisibility is the core concept behind modular arithmetic.

Definition: a divides b , written as $a|b$.

For $a \in \mathbb{Z}, b \in \mathbb{Z}, a|b \leftrightarrow \exists k \in \mathbb{Z}. b = ka.$

It all starts with divisibility ...

Divisibility is the core concept behind modular arithmetic.

Definition: a divides b, written as $a|b$.

For $a \in \mathbb{Z}, b \in \mathbb{Z}, a|b \leftrightarrow \exists k \in \mathbb{Z}. b = ka$.

Examples: which of the following are true and which are false?

$$5|1$$

$$25|5$$

$$5|0$$

$$2|3$$

$$1|5$$

$$5|25$$

$$0|5$$

$$3|2$$

It all starts with divisibility ...

Divisibility is the core concept behind modular arithmetic.

Definition: a divides b, written as $a|b$.

For $a \in \mathbb{Z}, b \in \mathbb{Z}, a|b \leftrightarrow \exists k \in \mathbb{Z}. b = ka$.

Examples: which of the following are true and which are false?

$$5|1 \quad \text{F}$$

$$25|5$$

$$5|0$$

$$2|3$$

$$1|5 \quad \text{T}$$

$$5|25$$

$$0|5$$

$$3|2$$

It all starts with divisibility ...

Divisibility is the core concept behind modular arithmetic.

Definition: a divides b, written as $a|b$.

For $a \in \mathbb{Z}, b \in \mathbb{Z}, a|b \leftrightarrow \exists k \in \mathbb{Z}. b = ka$.

Examples: which of the following are true and which are false?

$$5|1 \quad \text{F}$$

$$25|5 \quad \text{F}$$

$$5|0$$

$$2|3$$

$$1|5 \quad \text{T}$$

$$5|25 \quad \text{T}$$

$$0|5$$

$$3|2$$

It all starts with divisibility ...

Divisibility is the core concept behind modular arithmetic.

Definition: a divides b, written as $a|b$.

For $a \in \mathbb{Z}, b \in \mathbb{Z}, a|b \leftrightarrow \exists k \in \mathbb{Z}. b = ka$.

Examples: which of the following are true and which are false?

$$5|1 \quad \text{F}$$

$$25|5 \quad \text{F}$$

$$5|0 \quad \text{T}$$

$$2|3$$

$$1|5 \quad \text{T}$$

$$5|25 \quad \text{T}$$

$$0|5 \quad \text{F}$$

$$3|2$$

It all starts with divisibility ...

Divisibility is the core concept behind modular arithmetic.

Definition: a divides b , written as $a|b$.

For $a \in \mathbb{Z}, b \in \mathbb{Z}, a|b \leftrightarrow \exists k \in \mathbb{Z}. b = ka.$

Examples: which of the following are true and which are false?

$$5|1 \quad \text{F}$$

$$25|5 \quad \text{F}$$

$$5|0 \quad \text{T}$$

$$2|3 \quad \text{F}$$

$$1|5 \quad \text{T}$$

$$5|25 \quad \text{T}$$

$$0|5 \quad \text{F}$$

$$3|2 \quad \text{F}$$

Division theorem

Division theorem

For $a \in \mathbb{Z}$, $d \in \mathbb{Z}$ with $d > 0$,
there exist *unique* integers q, r with $0 \leq r < d$
such that $a = dq + r$.

If we divide a by d , we get a unique quotient $q = a \operatorname{div} d$ and non-negative remainder $r = a \operatorname{mod} d$.

Division theorem

Division theorem

For $a \in \mathbb{Z}, d \in \mathbb{Z}$ with $d > 0$,
there exist *unique* integers q, r with $0 \leq r < d$
such that $a = dq + r$.

If we divide a by d , we get a unique quotient $q = a \operatorname{div} d$ and non-negative remainder $r = a \operatorname{mod} d$.

Note that $r \geq 0$ even if $a < 0$, so **mod is not %**.

```
public class NotMod {
    public static void main(String args[]) {
        System.out.println("-5 mod 2 = 1.");
        System.out.println("-5 % 2 = " + (-5 % 2))
    }
}
```

```
$javac NotMod.java
$java -Xmx128M -Xms16M NotMod
-5 mod 2 = 1.
-5 % 2 = -1
```


Example: arithmetic mod 7

$$a +_7 b = (a + b) \bmod 7$$

+	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

$$a \times_7 b = (a \times b) \bmod 7$$

*	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

Modular arithmetic properties

Congruence, addition, multiplication, proofs.

Congruence modulo a positive integer

Definition: a is congruent to b modulo m , written as $a \equiv b \pmod{m}$

For $a, b, m \in \mathbb{Z}$ with $m > 0$, $a \equiv b \pmod{m} \leftrightarrow m \mid (a - b)$

Congruence modulo a positive integer

Definition: a is congruent to b modulo m , written as $a \equiv b \pmod{m}$

For $a, b, m \in \mathbb{Z}$ with $m > 0$, $a \equiv b \pmod{m} \leftrightarrow m \mid (a - b)$

Examples: what do these mean and when are they true?

$$x \equiv 0 \pmod{2}$$

$$-1 \equiv 19 \pmod{5}$$

$$y \equiv 2 \pmod{7}$$

Congruence modulo a positive integer

Definition: a is congruent to b modulo m , written as $a \equiv b \pmod{m}$

For $a, b, m \in \mathbb{Z}$ with $m > 0$, $a \equiv b \pmod{m} \leftrightarrow m \mid (a - b)$

Examples: what do these mean and when are they true?

$$x \equiv 0 \pmod{2}$$

True for every x that is divisible by 2, i.e., even.

$$-1 \equiv 19 \pmod{5}$$

$$y \equiv 2 \pmod{7}$$

Congruence modulo a positive integer

Definition: a is congruent to b modulo m , written as $a \equiv b \pmod{m}$

For $a, b, m \in \mathbb{Z}$ with $m > 0$, $a \equiv b \pmod{m} \leftrightarrow m \mid (a - b)$

Examples: what do these mean and when are they true?

$$x \equiv 0 \pmod{2}$$

True for every x that is divisible by 2, i.e., even.

$$-1 \equiv 19 \pmod{5}$$

True because $-1 - 19 = -20$ is divisible by 5.

$$y \equiv 2 \pmod{7}$$

Congruence modulo a positive integer

Definition: a is congruent to b modulo m , written as $a \equiv b \pmod{m}$

For $a, b, m \in \mathbb{Z}$ with $m > 0$, $a \equiv b \pmod{m} \leftrightarrow m \mid (a - b)$

Examples: what do these mean and when are they true?

$$x \equiv 0 \pmod{2}$$

True for every x that is divisible by 2, i.e., even.

$$-1 \equiv 19 \pmod{5}$$

True because $-1 - 19 = -20$ is divisible by 5.

$$y \equiv 2 \pmod{7}$$

True for every y of the form $y = 2 + 7k$ where $k \in \mathbb{Z}$.

Congruence and equality

Congruence property

Let $a, b, m \in \mathbb{Z}$ with $m > 0$.

Then, $a \equiv b \pmod{m}$ if and only if $a \bmod m = b \bmod m$.

Congruence and equality

Congruence property

Let $a, b, m \in \mathbb{Z}$ with $m > 0$.

Then, $a \equiv b \pmod{m}$ if and only if $a \bmod m = b \bmod m$.

Proof:

Suppose that $a \equiv b \pmod{m}$.

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

Congruence and equality

Congruence property

Let $a, b, m \in \mathbb{Z}$ with $m > 0$.

Then, $a \equiv b \pmod{m}$ if and only if $a \bmod m = b \bmod m$.

Proof:

Suppose that $a \equiv b \pmod{m}$.

Then $m \mid a - b$ by definition of congruence. So $a - b = km$ for some $k \in \mathbb{Z}$ by definition of divides. Therefore, $a = b + km$. Taking both sides modulo m , we get $a \bmod m = (b + km) \bmod m = b \bmod m$.

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

Congruence and equality

Congruence property

Let $a, b, m \in \mathbb{Z}$ with $m > 0$.

Then, $a \equiv b \pmod{m}$ if and only if $a \bmod m = b \bmod m$.

Proof:

Suppose that $a \equiv b \pmod{m}$.

Then $m \mid a - b$ by definition of congruence. So $a - b = km$ for some $k \in \mathbb{Z}$ by definition of divides. Therefore, $a = b + km$. Taking both sides modulo m , we get $a \bmod m = (b + km) \bmod m = b \bmod m$.

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

By the division theorem, $a = mq + (a \bmod m)$ and $b = ms + (b \bmod m)$ for some $q, s \in \mathbb{Z}$. Then, $a - b = (mq + (a \bmod m)) - (ms + (b \bmod m)) = m(q - s) + (a \bmod m - b \bmod m) = m(q - s)$, since $a \bmod m = b \bmod m$. Therefore, $m \mid (a - b)$ and so $a \equiv b \pmod{m}$.

The $\text{mod } m$ function vs the $\equiv (\text{mod } m)$ predicate

The $\text{mod } m$ function takes any $a \in \mathbb{Z}$ and maps it to a remainder $a \text{ mod } m \in \{0, 1, \dots, m - 1\}$.

In other words, $\text{mod } m$ places all integers that have the same remainder modulo m into the same “group” (a.k.a. “congruence class”).

The $\equiv (\text{mod } m)$ predicate compares $a, b \in \mathbb{Z}$ and returns true if and only if a and b are in the same group according to the $\text{mod } m$ function.

Modular addition property

Modular addition property

Let m be a positive integer ($m \in \mathbb{Z}$ with $m > 0$).

If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a + c \equiv b + d \pmod{m}$.

Modular addition property

Modular addition property

Let m be a positive integer ($m \in \mathbb{Z}$ with $m > 0$).

If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a + c \equiv b + d \pmod{m}$.

Proof:

Suppose that $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$. By the definition of congruence, there are k and j such that $a - b = km$ and $c - d = jm$. Adding these equations together, we get $(a + c) - (b + d) = m(j + k)$. Reapplying the definition of congruence, we get that $(a + c) \equiv (b + d) \pmod{m}$.

Modular multiplication property

Modular multiplication property

Let m be a positive integer ($m \in \mathbb{Z}$ with $m > 0$).

If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $ac \equiv bd \pmod{m}$.

Modular multiplication property

Modular multiplication property

Let m be a positive integer ($m \in \mathbb{Z}$ with $m > 0$).

If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $ac \equiv bd \pmod{m}$.

Proof:

Suppose that $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$. By the definition of congruence, there are k and j such that $a - b = km$ and $c - d = jm$.

So, $a = km + b$ and $c = jm + d$. Multiplying these equations together, we get $ac = (km + b)(jm + d) = kjm^2 + kmd + bjm + bd$.

Rearranging gives us $ac - bd = m(kjm + kd + bj)$. Reapplying the definition of congruence, we get that $ac \equiv bd \pmod{m}$.

Example: a proof using modular arithmetic

Let $n \in \mathbb{Z}$, and prove that $n^2 \equiv 0 \pmod{4}$ or $n^2 \equiv 1 \pmod{4}$.

Example: a proof using modular arithmetic

Let $n \in \mathbb{Z}$, and prove that $n^2 \equiv 0 \pmod{4}$ or $n^2 \equiv 1 \pmod{4}$.

Let's look at a few examples:

$$0^2 = 0 \equiv 0 \pmod{4}$$

$$1^2 = 1 \equiv 1 \pmod{4}$$

$$2^2 = 4 \equiv 0 \pmod{4}$$

$$3^2 = 9 \equiv 1 \pmod{4}$$

$$4^2 = 16 \equiv 0 \pmod{4}$$

Example: a proof using modular arithmetic

Let $n \in \mathbb{Z}$, and prove that $n^2 \equiv 0 \pmod{4}$ or $n^2 \equiv 1 \pmod{4}$.

Let's look at a few examples:

$$0^2 = 0 \equiv 0 \pmod{4}$$

$$1^2 = 1 \equiv 1 \pmod{4}$$

$$2^2 = 4 \equiv 0 \pmod{4}$$

$$3^2 = 9 \equiv 1 \pmod{4}$$

$$4^2 = 16 \equiv 0 \pmod{4}$$

It looks like

$$n \equiv 0 \pmod{2} \rightarrow n^2 \equiv 0 \pmod{4}$$

$$n \equiv 1 \pmod{2} \rightarrow n^2 \equiv 1 \pmod{4}$$

Example: a proof using modular arithmetic

Let $n \in \mathbb{Z}$, and prove that $n^2 \equiv 0 \pmod{4}$ or $n^2 \equiv 1 \pmod{4}$.

Let's look at a few examples:

$$0^2 = 0 \equiv 0 \pmod{4}$$

$$1^2 = 1 \equiv 1 \pmod{4}$$

$$2^2 = 4 \equiv 0 \pmod{4}$$

$$3^2 = 9 \equiv 1 \pmod{4}$$

$$4^2 = 16 \equiv 0 \pmod{4}$$

It looks like

$$n \equiv 0 \pmod{2} \rightarrow n^2 \equiv 0 \pmod{4}$$

$$n \equiv 1 \pmod{2} \rightarrow n^2 \equiv 1 \pmod{4}$$

Proof by cases:

Case 1 (n is even):

Case 2 (n is odd):

Example: a proof using modular arithmetic

Let $n \in \mathbb{Z}$, and prove that $n^2 \equiv 0 \pmod{4}$ or $n^2 \equiv 1 \pmod{4}$.

Let's look at a few examples:

$$0^2 = 0 \equiv 0 \pmod{4}$$

$$1^2 = 1 \equiv 1 \pmod{4}$$

$$2^2 = 4 \equiv 0 \pmod{4}$$

$$3^2 = 9 \equiv 1 \pmod{4}$$

$$4^2 = 16 \equiv 0 \pmod{4}$$

It looks like

$$n \equiv 0 \pmod{2} \rightarrow n^2 \equiv 0 \pmod{4}$$

$$n \equiv 1 \pmod{2} \rightarrow n^2 \equiv 1 \pmod{4}$$

Proof by cases:

Case 1 (n is even):

Suppose $n \equiv 0 \pmod{2}$. Then $n = 2k$ for some integer k . So $n^2 = (2k)^2 = 4k^2$. Therefore, by definition of congruence, $n^2 \equiv 0 \pmod{4}$.

Case 2 (n is odd):

Example: a proof using modular arithmetic

Let $n \in \mathbb{Z}$, and prove that $n^2 \equiv 0 \pmod{4}$ or $n^2 \equiv 1 \pmod{4}$.

Let's look at a few examples:

$$0^2 = 0 \equiv 0 \pmod{4}$$

$$1^2 = 1 \equiv 1 \pmod{4}$$

$$2^2 = 4 \equiv 0 \pmod{4}$$

$$3^2 = 9 \equiv 1 \pmod{4}$$

$$4^2 = 16 \equiv 0 \pmod{4}$$

It looks like

$$n \equiv 0 \pmod{2} \rightarrow n^2 \equiv 0 \pmod{4}$$

$$n \equiv 1 \pmod{2} \rightarrow n^2 \equiv 1 \pmod{4}$$

Proof by cases:

Case 1 (n is even):

Suppose $n \equiv 0 \pmod{2}$. Then $n = 2k$ for some integer k . So $n^2 = (2k)^2 = 4k^2$. Therefore, by definition of congruence, $n^2 \equiv 0 \pmod{4}$.

Case 2 (n is odd):

Suppose $n \equiv 1 \pmod{2}$. Then $n = 2k + 1$ for some integer k . So $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 4(k^2 + k) + 1$. Therefore, by definition of congruence, $n^2 \equiv 1 \pmod{4}$.

Modular arithmetic and integer representations

Unsigned, sign-magnitude, and two's complement representation.

Unsigned integer representation

Represent integer x as a sum of n powers of 2:

If $x = \sum_{i=0}^{n-1} b_i 2^i$ where each $b_i \in \{0, 1\}$,
then the representation is $b_{n-1} \dots b_2 b_1 b_0$.

Unsigned integer representation

Represent integer x as a sum of n powers of 2:

If $x = \sum_{i=0}^{n-1} b_i 2^i$ where each $b_i \in \{0, 1\}$,
then the representation is $b_{n-1} \dots b_2 b_1 b_0$.

Examples:

$$99 = 64 + 32 + 2 + 1$$

$$18 = 16 + 2$$

So for $n = 8$:

$$99 = 0110\ 0011$$

$$18 = 0001\ 0010$$

Unsigned integer representation

Represent integer x as a sum of n powers of 2:

If $x = \sum_{i=0}^{n-1} b_i 2^i$ where each $b_i \in \{0, 1\}$,
then the representation is $b_{n-1} \dots b_2 b_1 b_0$.

Examples:

$$99 = 64 + 32 + 2 + 1$$

$$18 = 16 + 2$$

So for $n = 8$:

$$99 = 0110\ 0011$$

$$18 = 0001\ 0010$$

This works for unsigned integers. How do we represent signed integers?

Sign-magnitude integer representation

If $-2^{n-1} < x < 2^{n-1}$, represent x with n bits as follows:

Use the first bit as the sign (0 for positive and 1 for negative), and the remaining $n - 1$ bits as the (unsigned) value.

Examples:

$$99 = 64 + 32 + 2 + 1$$

$$18 = 16 + 2$$

So for $n = 8$:

$$99 = 0110\ 0011$$

$$-18 = 1001\ 0010$$

Sign-magnitude integer representation

If $-2^{n-1} < x < 2^{n-1}$, represent x with n bits as follows:

Use the first bit as the sign (0 for positive and 1 for negative), and the remaining $n - 1$ bits as the (unsigned) value.

Examples:

$$99 = 64 + 32 + 2 + 1$$

$$18 = 16 + 2$$

So for $n = 8$:

$$99 = 0110\ 0011$$

$$-18 = 1001\ 0010$$

$$81 = 0101\ 0001$$

The problem with this representation is that our standard arithmetic algorithms no longer work, e.g., adding the representation of -18 and 99 doesn't give the representation of 81.

Two's complement integer representation

Represent x with n bits as follows:

If $0 \leq x < 2^{n-1}$, use the n -bit unsigned representation of x .

If $-2^{n-1} \leq x < 0$, use the n -bit unsigned representation of $2^n - |x|$.

Two's complement integer representation

Represent x with n bits as follows:

If $0 \leq x < 2^{n-1}$, use the n -bit unsigned representation of x .

If $-2^{n-1} \leq x < 0$, use the n -bit unsigned representation of $2^n - |x|$.

Key property:

Two's complement representation of any number y is equivalent to $y \bmod 2^n$ so arithmetic works $\bmod 2^n$.

Two's complement integer representation

Represent x with n bits as follows:

If $0 \leq x < 2^{n-1}$, use the n -bit unsigned representation of x .

If $-2^{n-1} \leq x < 0$, use the n -bit unsigned representation of $2^n - |x|$.

Key property:

Two's complement representation of any number y is equivalent to $y \bmod 2^n$ so arithmetic works $\bmod 2^n$.

Examples:

$$99 = 64 + 32 + 2 + 1$$

$$18 = 16 + 2$$

$$2^8 - 18 = 256 - 18 = 238 = 128 + 64 + 32 + 8 + 4 + 2$$

$$81 = 64 + 16 + 1$$

So for $n = 8$:

$$99 = 0110\ 0011$$

$$-18 = 1110\ 1110$$

$$81 = 0101\ 0001$$

Computing the two's complement representation

For $-2^{n-1} \leq x < 0$, x is represented using the n -bit unsigned representation of $2^n - |x|$.

Here is an easy way to compute this value:

- Compute the n -bit unsigned representation of $|x|$.
- Flip the bits of $|x|$ to get the representation of $2^n - 1 - |x|$.
 - This works because the string of all 1's represents $2^n - 1$.
- Add 1 to get $2^n - |x|$.

Applications of modular arithmetic

Hashing, pseudo-random numbers, ciphers.

Hashing

Problem:

We want to map a small number of data values from a large domain $\{0, 1, \dots, M - 1\}$ into a small set of locations $\{0, 1, \dots, n - 1\}$ to be able to quickly check if a value is present.

Solution:

Compute $\text{hash}(x) = x \bmod p$ for a prime p close to n .

Or, compute $\text{hash}(x) = ax + b \bmod p$ for a prime p close to n .

This approach depends on all of the bits of data the data.

Helps avoid collisions due to similar values.

But need to manage them if they occur.

Pseudo-random number generation

Linear Congruential method

$$x_{n+1} = (ax_n + c) \bmod m$$

Choose random x_0, a, c, m and produce a long sequences of x_n 's.

Simple ciphers

Cesar or shift cipher:

Treat letters as numbers: A = 0, B = 1, ...

$$f(p) = (p + k) \bmod 26$$

$$f^{-1}(p) = (p - k) \bmod 26$$

More general version:

$$f(p) = (ap + b) \bmod 26$$

$$f^{-1}(p) = (a^{-1}(p - b)) \bmod 26$$

Summary

Sets can be represented efficiently using bitvectors.

This representation is used heavily in the real world.

With this representation, set operations reduce to fast bitwise operations.

Modular arithmetic is arithmetic over a finite domain.

Key notions are divisibility and congruency $\text{mod } m$.

Modular arithmetic is the basis of computing.

Used with two's complement representation to implement computer arithmetic.

Also used in hashing, pseudo-random number generation, and cryptography.