CS 5002: Discrete Structures **Fall 2018**

Midterm Review: October 21, 2018 [1]

Instructors: Adrienne Slaughter, Tamara Bonaci

1 Week 1: Logic, Number Representations

1.1 Logic: Propositions, Operators, Logical Equivalence and Quantifiers

A ***proposition*** is defined as a statement that is true or false, but not both. Such a proposition has a **truth value**. The ***truth value*** of a proposition is true (**T**) if it is a true proposition and false (**F**) if it is false.

We introduced several logic operators:

- **Negation,** $\neg$ – the proposition $\neg p$ is false when p is true, and true when p is false.
- **Conjunction,** $\wedge$ – the proposition p and q is $p \wedge q$ and is *true* when p and q are true, and false otherwise.
- **Disjunction or inclusive OR,** $\vee$ – the proposition p or q (written by $p \vee q$) is *false* when p and q are false, and true otherwise.
- **Exclusive OR, or XOR,** $\oplus$ – the proposition p xor q (written by $p \oplus q$) is *true* when p is true OR q is true, but not when p and q are either both true or both false.
- **Implication or conditional,** $\rightarrow$ – the proposition $p \rightarrow q$ is false when p is true and q is false, and true otherwise.
- **Biconditional,** $\leftrightarrow$ – the proposition $p \leftrightarrow q$ is true when p is true and q is true, or p is false and q is false, false otherwise.

We can create ***compound propositions*** by grouping and ordering propositions. We can then evaluate ***logical equivalence*** of two or more such (compound) propositions by comparing the truth tables of each of them.

Predicates are statements that involve variables and cannot be determined to be true or false without specifying the value of the variable(s). ***Quantifiers*** help us express when a predicate is true over a range of values. The ***universe of discourse*** (or ***domain of discourse***) is the set of possible items the predicate applies to.

We pulled out two special quantifiers:

- ***Universal quantifier*** $\forall$: For every value in the *universe of discourse*, some predicate P is true.
- ***Existential quantifier*** $\exists$: There exists a value in the universe of discourse such that some predicate P is true.

1.2 Number Representations

In general we use *decimal* notation to express integers. However, we don't always have to use base 10. We can actually use any base, such as for example 5. There are general rules to represent some number n into

some arbitrary base b.

Theorem 1.1 *If b is a positive integer greater than 1, and n is a positive integer, it can be expressed in the form:*

$$n_{10} = a_k b^k + a_{k-1} b^{k-1} + ... + a_1 b + a_0$$

where k is a nonnegative integer, the number of digits in n

Some popular bases that we will be dealing with in computer sciences in general are:

- **Binary base** – base $b = 2$
- **Octal base** – base $b = 8$
- **Hexadecimal base** – base $b = 16$

To construct the base b expansion of integer n, first divide n by b to obtain a quotient and remainder. That is:

$$n = bq_0 + a_0, 0 \leq a_0 < b$$

The remainder (a_0) is the rightmost (least significant) digit in the base b expansion on n. Next, divide q_0 by b to obtain

$$q_0 = bq_1 + a_1, 0 \leq a_1 < b$$

a_1 is the second digit from the right in the base b expansion of n. Continue this until you obtain a quotient equal to 0.

We then briefly talked on the number of possible values that we can represent in some base. We observed that for some base b and digit length k, the maximum number of values that can be represented is b^k.

1.3 Representing Negative Numbers

We have only talked about positive numbers, but in real life, numbers can be negative. We indicate that by putting a negative sign in front of the number.

However, what do we do in machine world to represent negative numbers? There are several approaches: We can designate a particular bit, normally the left-most, to indicate the sign. One problem with this approach is that the procedures for adding the resulting numbers is somewhat complicated. Another is that the representation of zero is not unique.

One popular approach that resolves these two issues, is referred to as ***two's complement***. In this approach, the negative sign is encoded within the number representation itself.

Definition 1.2 (Two's Complement) *Given a positive integer a, the* **two's complement** *of a relative to the fixed bit length n is the n-bit binary representation of $2^n - a$.*

Two's complement - Explanation There is a convenient way to compute two's complements that involves less arithmetic then the direct application of the definition above. For example, let's consider an 8-bit representation. The representation is based on three facts:

1. $2^8 - a = [(2^8 - 1) - a] + 1$
2. The binary representation of $2^8 - 1$ is 11111111_2
3. Subtracting an 8-bit binary number a from 11111111_2 is equivalent to just switching all 0's to 1's, and vice versa. The resulting number is callend the **one's complement** of the given number.

Example 1: Find an 8-bit two's complement of $a = 27$. Start by representing $a = 27$ in the binary base as $a = 27_{10} = 00011011_2$:

$$\begin{array}{rl} & 11111111 = 2^8 - 1 \\ - & 00011011 = 27_2 \\ & 11100100 = (2^8 - 1) - 27 \\ + & 00000001 = 1 \\ \hline & 11100101 = 2^8 - 27 \end{array} \tag{1}$$

In general, to find an n-bit complement of a positive integer a:

1. Write an n-bit representation of a.
2. Flip the bits (that is, switch all 1's to 0's, and all 0's to 1's).
3. Add 1 to binary representation.

1.4 Logic in Computers

Definition 1.3 (Boolean variable) *A **Boolean variable** is a variable that can take on only two values, either true or false. Such a variable can be represented using a bit (binary digit).*

Computers represent information using **bits**, where a bit is defined as a symbol with two possible values, 0 (zero) and 1 (one). A bit can also be used to represent a truth value, and typically, we use a 1 bit to represent true, and a 0 bit to represent false.

This allows us to establish the correspondence between computer **bit operations**, as follows:

p	q	NOT p	p AND q	p OR q	p XOR q
1	1	0	1	1	0
1	0		0	1	1
0	1	1	0	1	1
0	0		0	0	0

One can now ask is: what if we have more than one bit of information to represent?

Once again, not a problem. We will use a **bit (binary) string**, defined as follows.

Definition 1.4 (Bit string) *A **bit string** is a sequence of zero or more bits, and its length is equal to the number of bits in this string.*

We can now use bit strings, and extend binary operations to bit strings as follows.

Bitwise OR, bitwise AND, and **BITWISE XOR** of two strings of the same length are the resulting bit strings, of the same length, that have as their bits the OR, AND or XOR of the corresponding bits in the starting strings.

Let's take a look at an example.

Example 2: Consider bit strings 0110 1101 1000 and 1100 0111 0110. Find the bitwise OR, bitwise AND and bitwise XOR of these strings.

$$\begin{array}{ll} 011011011000 & \\ 110001110110 & \\ \hline 111011111110 & (\text{bitwise } OR) \\ 000001010000 & (\text{bitwise } AND) \\ 111011101110 & (\text{bitwise } XOR) \end{array}$$

1.5 Vocabulary

Logic

- Proposition
- Truth value
- Truth table
- Logical operators
- negation
- conjunction
- disjunction
- xor
- inclusive or
- exclusive or
- conditional
- implication
- biconditional
- compound propositions
- logical equivalence
- predicates
- quantifiers
- universal quantifier
- existential quantifier
- universe of discourse
- using binary as T/F
- bitwise operations

Number Representations

- decimal
- fundamental theorem of arithmetic (?)
- base expansion
- binary expansion
- binary, hexadecimal, octal
- base b expansion
- value limits given number of digits
- 2's complement
- 1's complement
- max/min values of data types

2 Week 2: Variables and Functions

How does one express mathematical statements?

There exist several different ways to express mathematical statements. Three of the most important mathematical statements are:

- **Universal statements,** stating the property that is true for all elements in some set. (For example: *All students in this class are ALIGN students.*)
- **Conditional statement,** stating that if one thing is true, then it has to be the case that some other thing is also true. (For example *If a student is required to take CS 5002, then tht student is an ALIGN student.*)

- **Existential statement,** which states that, given some property, there exists at least one element of the set for which the given property is true. (For example: *It will be sunny at least one day this week.*)

Such mathematical statements can be expressed in a variety of different ways. We are interested in a way that is efficient and practical for computer systems. One such a way relies on variables and functions.

Definition 2.5 *A* **variable** *is any characteristic, number or quantity whose value:*

- Is arbitrary, or
- Is not fully specified, or
- Is unknow, or
- Changes over time

When thinking about variables, we typically distinguish between two types:

- **Independent variables** - variables that can take different values independent of other variables we may care to measure
- **Dependent variables** - variables that change value in relation (response) to other variables in our system

In algebra, a function is seen a relationship between one or more **inputs (input variables)**, and one or more **outputs (output variables)**.

$$\underbrace{f}_{\text{name of a function}}\left(\underbrace{x}_{\text{input variable}}\right) = \underbrace{y}_{\text{output variable}} \tag{2}$$

Definition 2.6 *Let A and B be some nonempty sets (sets containing at least one element). A function $f : A \to B$ from A to B is an assignment of exactly one element of B to each element of A. We write $f(a) = b$ if b is a unique element of B, assigned by the function f to the element $a \in A$.*

Definition 2.7 *If f is a function from A to B, we refer to A as a* **domain** *of f and B as a* **codomain**.

Definition 2.8 *If $f(a) = b$, we say that b is the* **image** *of a, and a is the* **preimage** *of b. The* **range** *or* **image** *of f is the set of all image of elements of A.*

Two functions are said to be **equal** if the have:

- The same domain,
- The same codomain, and
- Map each element of their common domain to the same element of their common codomain.

Definition 2.9 *Let f_1 and f_2 be functions from A to $\mathbb{R}$. Then $f_1 + f_2$ and $f_1 f_2$ are also functions from A to $\mathbb{R}$, defined for all $x \in A$ as follows:*

$$(f_1 + f_2)(x) = f_1(x) + f_2(x) \tag{3}$$
$$(f_1 f_2)(x) = f_1(x) f_2(x) \tag{4}$$

Definition 2.10 *Let f be a function from A to B, and let S be a subset from A. The image of S under the function f is the subset of B that consists of images of elements of S. We typically denote the image of S as $f(S)$*

$$f(S) = \{t | \exists s \in S(t = f(s))\} \tag{5}$$

Definition 2.11 *Some function f is said to be an* **injection** *or* **one-to-one** *if and only if $f(a) = f(b)$ implies that $a = b$ for all a and b in the domain of f.*

Remark: We can express that f is one-to-one using quantifiers:

$$\forall a \forall b(f(a) = f(b)) \rightarrow a = b$$

or equivalently:

$$\forall a \forall b(a \neq b \rightarrow f(a) \neq f(b))$$

where the universe of discourse if the domain of the function.

Definition 2.12 *Some function f is said to be an* **surjection** *or* **onto** *if and only if for every element $b \in B$ there is an element $a \in A$, such that $f(a) = b$.*

Definition 2.13 *Some function f is said to be an* **bijection** *or* **one-to-one correspondance** *if and only if it is both injection (one-to-one) and surjection (onto).*

Summary

- **To show that f is injective:** Show that if $f(x) = f(y)$ for arbitrary $x, y \in A$ with $x \neq y$, then $x = y$.
- **To show that f is not injective:** Find particular elements $x, y \in A$ such that $x \neq y$, but $f(x) = f(y)$.
- **To show that f is surjective:** Consider an arbitrary $y \in B$, and find an element x inA such that $f(x) = y$.
- **To show that f is not surjective:** Find a particular element $y \in B$ such that $f(x) \neq y \forall x \in A$.

Definition 2.14 *A function f whose domain and codomain are subsets of the set of real numbers is called:*

- **Increasing function**, *if $f(x) \geq f(y)$ whenever $x < y$ and x and y are both in the domain of f.*
- **Strictly increasing function**, *if $f(x) < f(y)$ whenever $x < y$ and x and y are both in the domain of f.*
- **Decreasing function**, *if $f(x) \leq f(y)$ whenever xy and x and y are both in the domain of f.*
- **Strictly decreasing function**, *if $f(x) > f(y)$ whenever $x < y$ and x and y are both in the domain of f.*

Definition 2.15 *Let f be a one-to-one correspondance (bijection) from some set A to some set B. The* **inverse function** *of f is the function that assigns to an element b belonging to B the unique element $a \in A$ such that $f(a) = b$. The inverse function of f is typically denoted as f^{-1}.*

Definition 2.16 *Let g be a function from the set A to the set B, and let f be a function from the set B to the set C. The composition of the functions f and g, denoted fro all $a \in A$ by $f \circ g$, is defined as:*

$$(f \circ g)(a) = f(g(a)) \tag{6}$$

Definition 2.17 *A* **partial function** *f from a set A to a set B is an assignment of each element a in a subset of A, called the* **domain of definition of** *f, of a unique element $b \in B$. The sets A and B are still called the domain and the codomain of f, but we say that f is* **undefined** *for elements in A that are not in the domain fo definition of f.*

The question of whether a partial function is a function or not frequently comes up. In mathematics, a partial function is NOT considered a function, but in computer science, a partial function IS considered a function. When asked whether a partial function is a function, your response should be: it depends. (Or, if you truly consider yourself a computer scientist, respond "Of course it is!" But know that if you're talking to a mathematician, you may have to prove them wrong).

2.1 Vocabulary

- variable
- independent variable
- dependent variable
- function
- input variable
- output variable
- identity function
- eraser function
- function as mapping or transformation
- domain
- codomain
- image
- preimage
- real-value function
- integer-value function
- one-to-one function
- onto function
- injection
- surjection
- bijection
- equal functions
- increasing function
- strictly increasing
- decreasing
- strictly decreasing
- function graph
- inverse function
- composition of functions
- floor function
- ceiling function

3 Sums and Sequences, Recurrences

In this section, we introduced the concepts of sums, sequences and recurrences. In particular, these three concepts allowed us to do things like understand and calculate values that build upon previous values, such as compounding interest.

Definition 3.18 *A **sequence** is a collection of ordered elements.*

We use a_n to denote the n^{th} term of the sequence.

In addition to a sequence being a collection of ordered numbers, we treat ***strings*** as sequences of ordered letters.

Definition 3.19 *A **geometric sequence** or **geometric progression** can be expressed as*

$$a, ar, ar^2, ar^3, \ldots, ar^n$$

where initial term a and ratio r are real numbers.

A geometric sequence is the discrete analog of the continuous exponential function: $f(x) = ar^x$.

Definition 3.20 *An **arithmetic sequence** or **arithmetic progression** can be expressed as:*

$$a, a + d, a + 3d, a + 4d, \ldots a + nd$$

where initial term a and difference d are real numbers.

An arithmetic sequence is the discrete analog of the continuous linear function: $f(x) = a + dx$.

Definition 3.21 *A **recurrence relation** for a sequence is one in which a given term a_n is defined in terms of one or more previous terms in the sequence.*

*A **sequence** is a **solution** of a recurrence if its terms satisfy the recurrence relation.*

A recurrence can have multiple solutions that satisfies it; however, there is only one solution to a recurrence given a specific ***initial condition***.

When we have a recurrence, we can either "provide a solution", which is determining a solution that satisfies the recurrence, or we can ***solve the recurrence***. *Solving* the recurrence requires finding the ***closed formula*** for a recurrence.

Definition 3.22 *The **closed formula** of a recurrence is an expression of the recurrence solely in terms of n.*

The closed form of a recurrence can not be expressed with previous terms in the sequence; it must be expressed only in terms of the current term, n. We cannot determine the closed formula for a recurrence without the initial condition(s) being specified.

We use ***iteration*** or ***substitution*** to determine the closed form. When performing this process, we can either go ***forward*** or ***backward***.

To perform forward substitution, we start with a_0. We then write down a_1 according to the recurrence (which will be defined in terms of a_0, or the initial condition(s)), and then we substitute in a_0. Next, write down a_2 according to the recurrence, and substitute in a_1 that we determined in the previous step, which is already expressed in terms of a_0. Continue this process until you see a pattern, and derive the close form.

Backward substitution is the same process, except we start with a_n. Write down a_n according to the recurrence. Then, re-write a_n replacing the a_{n-1} terms with the equivalent according to the recurrence. Then, re-write a_n replacing the a_{n-2} terms. Continue until you see a pattern. In this approach, at each step, you write a_n in terms of only a single previous term, such as a_{n-3}.

Definition 3.23 (Summation Notation) ***Summation notation*** *allows us to express the sum of the terms $a_m, a_{m+1}, a_{m+2}, \ldots a_n$ as*

$$\sum_{j=m}^{n} a_j \Rightarrow a_m + a_{m+1} + a_{m+2} \ldots + a_n \tag{7}$$

Summations also usually have a closed form, but we did not cover how to derive the closed form of a summation. We can, however, use existing tables to look up the closed form of common summations. To do this, we need to be aware of the indices of summation: make sure that the indexes of the original summation match the indices of the form we're comparing to. You may have to shift the summation a bit:

$$\sum_{j=1}^{5} j \Leftrightarrow \sum_{k=0}^{4} (k+1) \tag{8}$$

Another helpful approach may be to pull out the first term:

$$\sum_{j=a}^{n} f(j) = f(a) + \sum_{j=a+1}^{n} f(j) \tag{9}$$

3.1 Vocabulary

- sequence
- summation
- geometric sequence
- arithmetic sequence
- recurrence relation
- closed formula
- iteration
- substitution
- initial condition
- fibonacci sequence

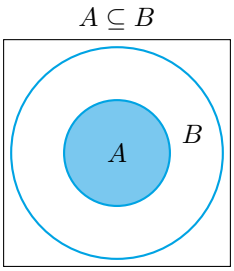
4 Sets and Matrices

A ***set*** is a group of objects, usually with some relationship or similar property. The objects in the set are called ***elements*** or ***members*** of the set. We use the symbol $\in$ to indicate that an element is or is not in a set:

$$x \in A\text{: } x \text{ is in set } A$$

$$x \notin A\text{: } x \text{ is not in set } A$$

Definition 4.24 (Venn Diagrams) *A **Venn diagram** is a graphical representation of a set.*



Definition 4.25 (Subsets) *The set A is a **subset** of B if and only if every element of A is also an element of the set B. We use the notation: $A \subseteq B$.*

Definition 4.26 (Set Cardinality) *Let S be a set. If there are n distinct elements in S (and n is an integer greater than or equal to 0), S is a **finite set**, and n is the **cardinality** or S. The cardinality of S is written $|S|$.*

Set Operations

- ***Union:*** Let A and B be sets. The **union** of the sets A and B, denoted $A \cup B$ is the set that contains the elements in either A or in B, or in both.
- ***Intersection:*** Let A and B be sets. The **intersection** of the sets A and B, denoted $A \cap B$ is the set that contains the elements in both A and B.
- ***Set complement:*** The **complement** of a set A, denoted A^c is the set of elements that belong to U but which do not belong to A.
- ***Difference of sets:*** The **relative complement** or **difference** of a set B with respect to A, denoted $A \setminus B$ (said A minus B) is the set of elements that belong to A, but which do not belong to B.

4.1 Matrices and Arrays

Last lecture, we defined a ***matrix*** as a rectangular array of numbers, and we said that a matrix with m rows and n columns is called an $m \times n$ matrix. Let's recall some matrix operation and properties we mentioned last time.

Matrix Operations and Properties

An ***identity matrix***, denoted as $I \in \mathbb{R}^{n \times n}$ is a square matrix with ones on the diagonal and zeros everywhere else:

$$I_{ij} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$$

A ***diagonal matrix*** is a matrix where all off-diagonal elements are equal to 0, $D = diag(d_1, d_2, \ldots, d_n)$ and

$$D_{ij} = \begin{cases} d_i, i = j \\ 0, i \neq j \end{cases}$$

Note: $I = diag(1, 1, \ldots, 1)$.

Given a matrix $A \in \mathbb{Z}^{m \times n}$, its ***transpose***, denoted $A^T \in \mathbb{Z}^{n \times m}$, is the $n \times m$ matrix whose entries are obtained by "flipping" the rows and the columns of matrix A:

$$(A^T)_{ij} = A_{ji}$$

The following properties of transpose can be easily verified:

- $(A^T)^T = A$
- $(AB)^T = B^T A^T$
- $(A + B)^T = A^T + B^T$

A **square matrix** $A \in \mathbb{Z}^{n \times n}$ is a ***symmetric matrix*** if it holds that:

$$A = A^T$$

A square matrix A is ***anti-symmetric*** if:

$$A = -A^T$$

Definition 4.27 (Matrix Manipulation) *The product of two matrices $A \in \mathbb{Z}^{m \times n}$ and $B \in \mathbb{Z}^{n \times p}$ is a new matrix, C, defined as:*

$$C = AB := \sum_{k=1}^{n} A_{ik} B_{kj} \in \mathbb{R}^{m \times p}$$

Note: *In order for matrix C to exist, the number of columns of matrix A must be equal to the number of rows of matrix B.*

Example 1: Please find matrix product AB, if matrices A and B are equal to $A = \begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & 4 & 2 \\ 1 & 5 & 3 \end{bmatrix}$.

$$AB = \begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 1 & 5 & 3 \end{bmatrix} = \begin{bmatrix} 2\cdot 2+3\cdot 1 & 2\cdot 4+3\cdot 5 & 2\cdot 2+3\cdot 3 \\ 4\cdot 2+1\cdot 1 & 4\cdot 4+1\cdot 5 & 4\cdot 2+1\cdot 3 \end{bmatrix} = \begin{bmatrix} 7 & 23 & 13 \\ 8 & 21 & 11 \end{bmatrix}$$

Properties of Matrix Manipulation:

- Matrix manipulation is associative: $(AB)C = A(BC)$
- Matrix manipulation is distributive: $A(B + C) = AB + Ac$
- Matrix manipulation in general is **not commutative**, i.e., in general, $AB \neq BA$

Definition 4.28 (Matrix Inverse) *The* **inverse** *of a square matrix $A \in \mathbb{Z}^{n\times n}$ is denoted as A^{-1}, and it is a* **unique** *matrix such that:*

$$A^{-1}A = AA^{-1} = I \tag{10}$$

Note: *Not all matrices have an inverse. In particular, we say that matrix A is invertible or* **non-singular** *if its inverse A^{-1} exits.*

4.2 Vocabulary

- set
- elements
- members
- set builder notation
- $\mathbb{N}, \mathbb{R}, \ldots$
- universal set
- Venn diagram
- subset
- proper subset
- equality of sets
- finite set
- infinite set
- cardinality
- power set
- empty set
- n-tuple
- Ordered n-tuple
- ordered pair
- cartesian product
- union
- intersection
- disjoint
- inclusion-exclusion principle
- complement
- difference
- relative complement
- symmetric difference
- set operations applied to many sets
- membership table
- DeMorgan's law
- set identities
- matrix
- matrix addition
- scalar multiplication
- matrix multiplication
- transpose
- vectors
- zero-one matrix
- column vector
- row vector
-

5 Number Theory

Last time, we introduce the notion of divisibility, and we said that, if a and b are integers such that $a \neq 0$, we say that **a divides b** if if there exist an integer c such that:

$$b = ac$$

or equivalently, $\frac{b}{a}$ is an integer. We introduce **the division algorithm** as follows.

Let a be some integer, and d some positive integer. Then there always exist unique integers q and r, where $0 \leq r < d$, such that:

$$a = dq + r \tag{11}$$

In equation (11), positive integer d is typically referred to as **divisor**, integer a as **dividend**, and integers q and r as **quotient** and **remainder**, respectively.

We next introduced **congruences**, and defined them as follows.

Let a and b be integers, $a, b \in \mathbb{Z}$ and let m be a positive integer, $m \in \mathbb{N}$. If m divides $(a - b)$, we can write:

$$a \equiv b \bmod m, \text{ or} \tag{12}$$
$$m|(a - b) \tag{13}$$

The operator $\equiv$ is called *congruence* and $a \equiv b \bmod m$ is read: "a **is congruent to** b **modulo** m." The positive integer m is known as the **modulus**.

We next defined **prime numbers** as those integers greater than one, whose only positive factors are 1 and p, and we answered the following questions about prime numbers:

- How do we show that some positive integer is a prime? **Trivial division**
- If an integer isn't a prime, how do we find all of its divisors (factors)? **Unique prime factorization**
- How many primes are there anyway? **Infinitely many**

We further introduced the concept of a **greatest common divisor** as follows:

Given two integers $a \neq 0$ and $b \neq 0$, the **greatest common divisor** of a and b (denoted $\gcd(a, b)$) is equal to the largest integer c that divides both a and b.

We defined that two integers $a \geq 1$ and $m \geq 2$ are said to be **relatively prime** or **coprime** if their greatest common divisor is equal to $gcd(a, m) = 1$.

We then answered the following questions?

- Given some positive integer a, how many integers from the set $\mathbb{Z}_a = \{1, 2, \ldots, a - 1\}$ are coprime with a? **Euler's totient function**
- How would we generally check whether or not two integers a and b are coprime? **Eucliedan algorithm**

We showed that fact that the greatest common divisor of some integers a and b can be expressed as a **linear combination of** a **and** b, and their corresponding linear coefficients

$$ax + by = d \tag{14}$$

where integers x and y are called **Bezout's coefficients**.

We further defined a **linear congruence** as a congruence of the form:

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

where m is a positive integer, a and b are integers, and x is a variable, and we showed how and when can we solve such linear congruences.

To do so, we introduce the notion of a **modular multiplicative inverse** of an integer $a \in \mathbb{Z}_m$ modulo m, denoted as $a^{-1} \bmod m$, as an element $a' \in \mathbb{Z}_m$ such that:

$$a \cdot a' \equiv a' \cdot a \equiv 1 \bmod m \tag{15}$$

And we showed that, given the modular multiplicative inverse, some congruence $ax \equiv b \bmod m$ can be solved for x as follows:

$$\begin{aligned} ax &\equiv b \bmod m \\ \underbrace{a^{-1}(ax)}_{x} &\equiv a^{-1}(b) \bmod m \end{aligned}$$

We answered the following questions about modular multiplicative inverses:

- When does an integer $a \in \mathbb{Z}_m$ have a modular multiplicative inverse under modulo m?
- If an integer $a \in \mathbb{Z}_m$ have a modular multiplicative inverse under modulo m, how do we find it? **Extended Euclidean algorithm**

Theorem 5.29 (Fermat's Little Theorem) *Let's consider two integers a and p. If p is a prime, and p does not divide a, then:*

$$a^{p-1} = 1 \bmod p \tag{16}$$

5.1 Vocabulary

- divides
- factor
- divisor
- multiple
- div
- mod
- quotient
- remainder
- congruent
- modulo
- modulus
- prime
- composite
- fundamental theorem of arithmetic
- prime factorization
- coprime
- relatively prime
- greatest common divisor/gcd
- least common multiple/lcm
- Euler-phi
- Euclidean algorithm
- Bezout's theorem
- linear congruence
- Euler's theorem
- Fermat's little theorem
- Chinese Remainder theorem

Math symbols

- $\equiv$
- $\bmod m$
- $a|b$
- $a \not| b$
- $a^{\phi(n)}$
- $\mathbb{Z}_a$

6 Relations

Relations indicate the relationship between two or more sets. When the relation applies to two sets, it's called ***binary relation***; if it's more than 2, it's a ***n-ary relation***.

Definition 6.30 (Binary Relation) *A and B are sets. A **binary relation** from A to B is a subset of the Cartesian product $A \times B$.*

The binary relation is a set of ordered pairs where the first element is in set A, and the second element is in set B. We write a binary relation as: aRb that means ordered pair $(a, b) \in R$. We can also write $a \not R b$, which means ordered pair $(a, b) \notin R$.

We particularly use relations that are from some set A to itself:

Definition 6.31 (Relation on a set) *A **relation on a set** is a relation from set A to set A.*

For example, we may have a relation on $\mathbb{Z}$ or $\mathbb{Z}R\mathbb{Z}$, which means a relation on the integers to the integers.

Functions are a special kind of relation.

Relations can have specific properties:

Definition 6.32 (*Reflexivity*) *A relation R on a set A is called **reflexive** if $(a, a) \in R$ for every element $a \in A$.*

Definition 6.33 (*Symmetry*) *A relation R on a set A is called **symmetric** if $(b, a) \in R$ whenever $(a, b) \in R$ for all $a, b \in A$.*

Definition 6.34 (*Anti-symmetry*) *A relation R on a set A is called **anti-symmetric** when it holds that for all $a, b \in A$ such that $(a, b) \in R$, if $(b, a) \in R$, then it follows that $a = b$.*

Definition 6.35 (*Transitivity*) *A relation R on a set A is called **transitive** if whenever $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$, for all $a, b, c \in A$.*

More informally, reflexivity, symmetry and transitivity properties say the following:

- **Reflexivity:** Each element of a set is related to itself.
- **Symmetry:** If any one element is related to any other element, then the second element is also related to the first.
- **Transitivity:** If any one element is related to a second, and that second element is related to some third element, then the first element is related to the third element too.

Or:

- R is reflexive $\iff \forall x \in A, (x, x) \in R$
- S is symmetric $\iff \forall x, y \in A,$ if $(x, y) \in R,$ then $(y, x) \in R$
- R is transitive $\iff \forall x, y, z \in A,$ such that $(x, y) \in R$ and $(y, z) \in R$, then $(x, z) \in R$

6.1 Vocabulary

- relation
- binary relations
- n-ary relations
- Cartesian product
- relation properties
- reflexive
- symmeteric
- anti-symmetric
- transitive
- composite relations
- zero-one matrix
- closure
- symmetric closure
- transitive closure
- reflexive closure
- equivalence relation
- equivalence class
- partial ordering
- comparable elements
- total order
- linear order
- chain
- well-ordered set
- lexicographic order

Symbols

- $\preceq$
- $A \ R \ B$
- aRb