CMPT-335 Discrete Structures

THE INTEGERS AND DIVISION. MODULAR ARITHMETIC

Encryption

- In the modern word, it is crucial that the information is transmitted and stored safely.
- For example, Internet purchases, bank and credit card electronic transactions must be secure, which means that no confidential information can be lost.
- To protect any information, encryption must be used.

Encryption: Mathematical Background

- The most popular techniques of encryption are based on the specific operations defined on the set of integer numbers and on some specific properties of integer numbers, which are studied in detail in the Number Theory.
- Here we will consider some of these properties and operations

Divisors

- Let a, b and c be integers ($a \neq 0$)such that

$$b = a \cdot c$$

- Then a and c are said to divide b (or are factors of b), while b is said to be a multiple of a and c.
- The notation is $a|b$, $c|b$ ("a divides b", "c divides b")

Divisor Theorem

- Theorem: Let a, b, and c be integers. Then:

1. *If* a divides b and a divides c *then* a divides $(b + c)$
2. *If* a divides b, *then* a divides bc for any c.
3. *If* a divides b, and b divides c, *then* a divides c.

Division Algorithm

- If *m* and *n* are integers and $m \neq 0$, the division algorithm states that *n* can be expressed in the form

$$n = qm + r, \quad 0 \leq r < |m|$$

- *q* is called quotient and *r* is called remainder.
- Take into account that the **remainder must always be non-negative**!

Division Algorithm

- $7:2 \rightarrow 7 = 3 \cdot 2 + 1$

  quotient remainder

- $-7:2 \rightarrow -4 \cdot 2 + 1$

  quotient remainder

Prime Numbers

- A number $n \geq 2$ is called a prime number if it is only divisible by 1 and itself. A number $n \geq 2$ which isn't prime is called composite.

Modular Arithmetic: mod n

- The mod *n* function:
  x **mod** *n* is the remainder of *x*/*n*.

Modular Arithmetic: mod n

- Definition of $f(x) = x \bmod n$ function, $x \in Z^{+}$

$$f(x) = \begin{cases} x, 0 \le x < n \\ r; x = qn + r, x > n \end{cases}$$

- In general, $f(x) = x \bmod n$ function, $x \in Z$

$$f(x) = r; x = qn + r$$

- Codomain of these functions is {0, 1, ..., n-1}.

Modular Arithmetic: Congruence

- The congruence (mod m)
  - Relates two numbers x and y to each other with respect to the base m
  - $x \equiv y \pmod{m}$ means that x and y have the same remainder when dividing by m.
- Let x, y be integers and m be a positive integer. We say that
  - x is congruent to y modulo m: $x \equiv y \pmod{m}$ if m divides $(x - y)$

Congruence

- Which of the following are true?

1. $3 \equiv 3 \pmod{17}$
2. $4 \equiv -4 \pmod{17}$
3. $182 \equiv 187 \pmod{5}$
4. $-15 \equiv 15 \pmod{30}$

Congruence

- Solutions:

1. $3 \equiv 3 \pmod{17}$ True. any number is congruent to itself (3-3 = 0, divisible by all)
2. $4 \equiv -4 \pmod{17}$ False. (4-(-4)) = 8 isn't divisible by 17.
3. $182 \equiv 187 \pmod{5}$ True. 182-187 = -5 is divisible by 5
4. $-15 \equiv 15 \pmod{30}$ True: -15-15 = -30 divisible by 30.

Congruence

- Theorem. x is congruent to y (mod m) when $x=y+km$ for some integer k.
- Corollary 1. x is congruent to the remainder in the division of x by m.
- Corollary 2. x is congruent to y (mod m) if and only if x and y have the same remainder when divided by m.

Congruence Classes

- Congruence mod *m* establishes the equivalence relation on the set *Z* of integer numbers because it is reflexive, symmetric, and transitive.
- The equivalence classes for congruence mod *m* are called congruence classes (mod *m*). The set of all congruence classes mod *m* is denoted Z_m.

Congruence Classes

- Since any two equivalence classes are either equal or disjoint, then any two congruence classes mod m are either equal or disjoint.
- Just to recall: $[x]$ is the equivalence class determined by the element x.
- In Z_m , $[x] = [y]$ if and only if $x \equiv y \pmod{m}$.

Congruence Classes

- If r is the remainder in the division of x by m, then $[x] = [r]$ in Z_m .
- Hence, there are m distinct congruence classes in Z_m : $[0], [1], \ldots, [m\text{-}1]$ corresponding to the m possible remainders when dividing by m.

Congruence Classes

- For example, there are 3 distinct equivalence classes in Z_3 : [0]= {…, -6, -3, **0**, 3, 6, …}; [1]= {…,-5, -2, **1**, 4, 7, …}; [2]= {…,-4, -1, **2**, 5, 8, …}
- Each congruence class can be determined by its arbitrary representative. It is just more convenient, to avoid any confusion, to use "natural" notations [0], [1], …, [m-1] for the distinct congruence classes in Z_m.

Congruence Classes

- In Z_m congruence classes are:
  [0]= {…,0-3m, 0-2m, 0-m, **0**, 0+m, 0+2m, 0+3m,…};
  [1]= {…,1-3m, 1-2m, 1-m, **1**, 1+m, 1+2m, 1+3m,…};
  [2]= {…,2-3m, 2-2m, 2-m, **2**, 2+m, 2+2m, 2+3m,…};
  …
  [m-1]={…,(m-1)-m=-1,(**m-1**),(m-1)+m,…}

Operations in Z_m

- Theorem. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then
  - $a + c \equiv b + d \pmod{m}$
  - $ac \equiv bd \pmod{m}$
- Corollary. If $x \equiv z \pmod{m}$ then

$$\forall n > 0 : x^n \equiv z^n \left(\bmod m\right)$$

and moreover, $\forall n > 0 : \left[x\right]^n = \left[z\right]^n$

Operations in Z_m

- Corollary. Let m be a positive integer and let x and y be integers. Then

$$(x+y) \bmod m = ((x \bmod m) + (y \bmod m)) \bmod m$$

$$xy \bmod m = ((x \bmod m)(y \bmod m)) \bmod m$$

Operations in Z_m

- Addition: $[x]+[y]=[x+y]$
- Multiplication: $[x][y]=[xy]$
- These definitions depend only on congruence classes themselves, but not on their particular representatives.
- This means that any arithmetic operation on the congruence class can be done with any representative of this class and the result does not depend on those classes' representatives involved

Operations in Z_m

- Examples of operations:
- In Z_8 : $[5]+[3] =$
- In Z_8 : $[5][3] =$
- In Z_8 : $[7]^5 =$

Operations in Z_m

- Examples of operations:
- In Z_8 : $[5]+[3] = [5 + 3] = [8] = [0]$ since $8 \equiv 0 \pmod 8$
- In Z_8 : $[5][3] = [5 \cdot 3] = [15] = [7]$ since $15 \equiv 7 \pmod 8$
- In Z_8 : $[7]^5 = [-1]^5 = [(-1)^5] = [-1] = [7]$ since $7 \equiv -1 \pmod 8$.

Operations in Z_m

- Examples of operations:
- In Z_8 : $[180]+[300]=$
- In Z_8 : $[180][300] =$
- In Z_8 : $[559]^5=$

Operations in Z_m

- Examples of operations:
- In Z_8 : $[180]+[300] = [180 \bmod 8+300 \bmod 8]=$
  $=[4+4] = [8] = [0]$ since $8\equiv 0 \pmod 8$
- In Z_8 : $[180][300] = [4\cdot 4]= [16] = [0]$
  since $16\equiv 0 \pmod 8$
- In Z_8 : $[559]^5 =[559 \bmod 8]^5=[-1]^5=[(-1)^5]=[-1]=[7]$
  since $7\equiv -1 \pmod 8$.

Operations in Z_m

- Example. A power generator was fully fueled at 9-00 am. It uses 2 gallons/hour of fuel. The capacity of a tank is 60 gallons. When it will be necessary to fuel the generator again?
- Let 0=12am, 1= 1am, …, 12=12pm, 13=1pm, …, 23 = 11pm. Then in Z_{24} :

  [9]+[60/2] = [9] + [30] = [9+30] = [39] = [15] since $39 \equiv 15 \pmod{24}$. Since 15 corresponds to 3pm, this is the next time of fueling (3pm next day).

Applications: Hashing Functions

- Suppose we have a huge database of personal data (at insurance company, bank, hospital, big university, big company, etc.)
- Each record has its unique key (for example, the Social Security Number)
- How can indexes or memory locations be assigned so that the database records can be retrieved quickly?

Applications: Hashing Functions

- A hashing function h assigns index or memory location $h(k)$ to the record that has k as its key:

$$h(k) = k \bmod m$$

where m is the number of available indexes or memory locations

Applications: Hashing Functions

- It may happen that the index or memory location

$$h(k) = k \bmod m$$

is already reserved for the record with the key *p* such that $k \equiv p \bmod m$ In this case the hashing function has to be equal to $h(k) = (k \bmod m) + 1$ or to the next available value of index (the next available memory location)

Applications: Generation of Pseudorandom Numbers

- Uniformly distributed random numbers are often needed for computer simulations of real-world processes
- What stands behind the “random” function in high level programming languages?

Applications: Generation of Pseudorandom Numbers

- The most commonly used algorithm for their generation is linear congruential algorithm
- This algorithm depends of the following parameters: modules m, multiplier a, increment c and seed x_0 :

$$x_1 = (ax_0 + c) \bmod m$$

$$\ldots$$

$$x_{i+1} = (ax_i + c) \bmod m$$

$$\ldots$$

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

Applications: Generation of Pseudorandom Numbers

- To generate pseudorandom numbers located in the interval [0,1], they should be normalized:

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

- This algorithm allows to generate ***m*** different random numbers before repetition begins
- In the modern high level languages environments where integers are 32-bit numbers, $m = 2^{31} - 1; a = 7^5 = 16807$ are used