Integers & Division (Lecture – 2)

Dr. Nirnay Ghosh

GCDs as Linear Combinations

- GCD *(a, b)* can be expressed as a **linear combination** with integer coefficients of a and b.
  - For example, gcd $(6, 14) = 2$, and $2 = (-2)*6 + 1*14$.
- **BÉZOUT'S THEOREM:** If a and b are positive integers, then there exist integers s and t such that $\gcd(a, b) = sa + tb$.
- **Definition:** If a and b are positive integers, then integers s and t such that $gcd(a, b) = sa + tb$ are called *Bézout coefficients* of a and b. Also, the equation $gcd(a, b) = sa + tb$ is called *Bézout's identity*.
- General Method to find linear combination of two integers equal to their gcd:
  - Proceed by working backward through the divisions of the Euclidean algorithm
  - Requires a forward pass and a backward pass through the steps of the Euclidean algorithm
- **LEMMA**: If a, b, and c are positive integers such that $gcd(a, b) = 1$ and $a \mid bc$, then $a \mid c$.

Unique Factorization of Integers

- Every integer can be written as the product of primes in non-decreasing order in at most one way.

Theorem 4.3.5 Unique Factorization of Integers Theorem (Fundamental Theorem of Arithmetic)

Given any integer $n > 1$, there exist a positive integer k, distinct prime numbers $p_1, p_2, \ldots, p_k$, and positive integers $e_1, e_2, \ldots, e_k$ such that

$$n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \ldots p_k^{e_k},$$

and any other expression for n as a product of prime numbers is identical to this except, perhaps, for the order in which the factors are written.

- **<u>LEMMA</u>:** If p is a prime and $p \mid a_1 a_2 \ldots a_n$, where each a_i is an integer, then $p \mid a_i$ for some i.
  - Lemma can be used to show that a factorization of an integer into primes is unique.

Modular Arithmetic

- In computer science we often care about the remainder of an integer when it is divided by some positive integer.
- **Problem**: Assume that it is a midnight. What is the time on the 24 hour clock after 50 hours?
- **Answer**: Its 2 AM.
  - How did we arrive to the result: Divide 50 with 24. The reminder is the time on the 24 hour clock
    - 50= 2*24 + 2
    - so the result is 2am.

Modular Arithmetic/Congruency

- **Definition**: If a and b are integers and m is a positive integer, then **a is congruent to b modulo m** if m divides (a-b). We use the notation **a ≡ b (mod m)** to denote the congruency. If a and b are not congruent we write a ≠ b (mod m).
- **Theorem #1**: If a and b are integers and m a positive integer. Then $a \equiv b \ (mod\ m)$ if and only if $a \bmod m = b \bmod m$.
- **Theorem #2**: Let m be a positive integer. The integers a and b are congruent modulo m if and only if there is an integer k such that $a = b + km$.
- **Theorem #3**: Let m be a positive integer. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a + c \equiv b + d \pmod{m}$ and $ac \equiv bd \pmod{m}$.

Arithmetic Modulo-m

- We can define arithmetic operations on $\mathbf{Z}_m$, the set of nonnegative integers less than m, that is, the set $\{0, 1, \ldots, m-1\}$.
- The addition of these integers, denoted by $+_m$ *(addition modulo-m)*, is given as:

$$a +_m b = (a + b) \bmod m,$$

where the addition on the right hand side of this equation is the ordinary addition of integers.

- The multiplication of these integers, denoted by $\cdot_m$ (*multiplication modulo-m*) is given as:

$$a \cdot_m b = (a \cdot b) \bmod m,$$

where the multiplication on the right-hand side of this equation is the ordinary multiplication of integers.

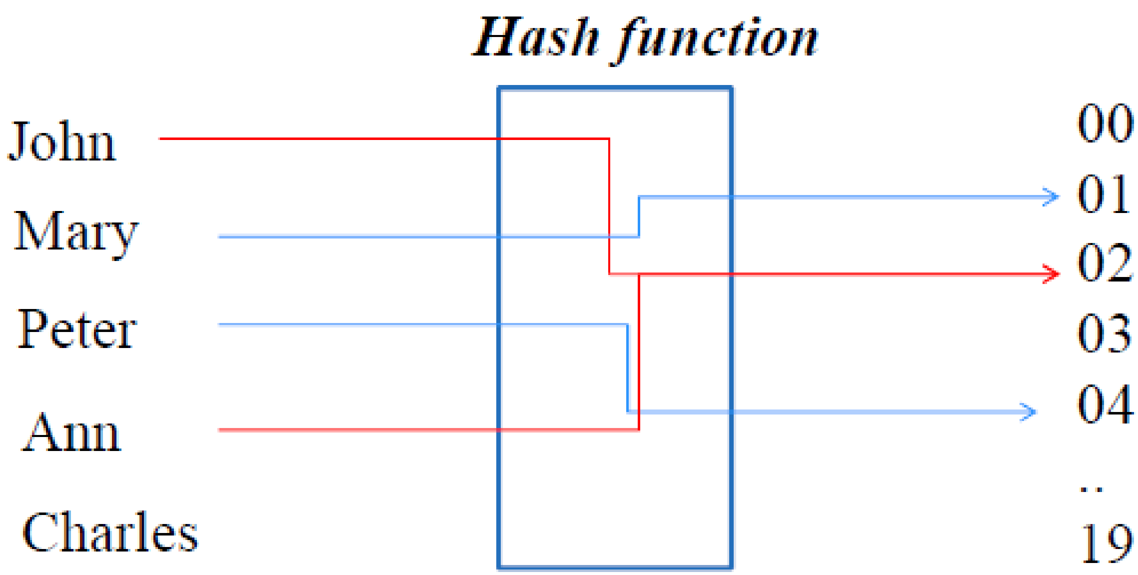
Application of Modular Arithmetic (1)

- **Pseudorandom number generators**: randomly chosen numbers are often needed for computer simulations.
  - Basic problem:
    - Assume outcomes: *0, 1, .. N*
    - Generate the random sequences of outcomes
  - Because numbers generated by systematic methods are not truly random, they are called ***pseudorandom numbers***.
- The most commonly used procedure for generating pseudorandom numbers is the **linear congruential method**.
- We choose four integers: the **modulus** m, **multiplier** a, **increment** c, and **seed** x_0 with $2 \leq a < m$, $0 \leq c < m$, and $0 \leq x_0 < m$.
- We generate a sequence of pseudorandom numbers $\{x_n\}$, with $0 \leq x_n < m$ for all n, by successively using the recursively defined function:

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

Application of Modular Arithmetic (2)

- **Hash Functions**:
  - A ***hash function*** is an algorithm that maps data of arbitrary length (also known as *keys*) to data of a fixed length
  - The values returned by a hash function are called **hash values** or **hash codes.**



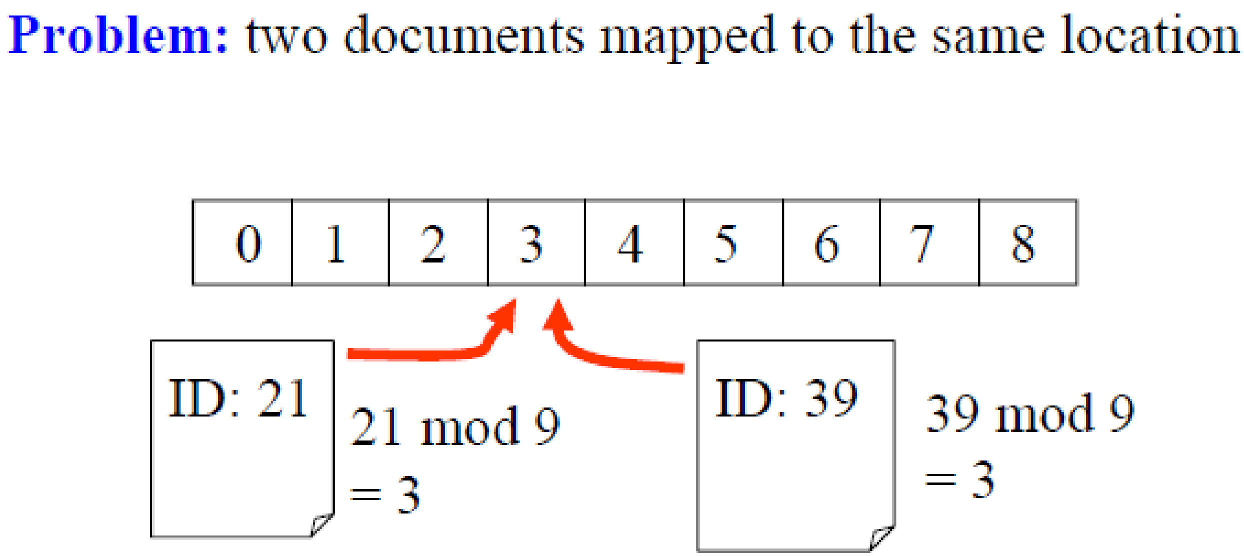
Application of Modular Arithmetic (2)

- <u>Problem</u>: Given a large collection of records, how can we store and find records quickly?
- <u>Solution</u>: Use a hash function calculate the location of the record based on the record's ID. A common hash function is $\boldsymbol{h(k) = k \bmod n}$, where n is the number of available storage locations.
- <u>Example</u>: Assume we have a database of employees, each with a unique ID - a social security number that consists of 8 digits. We want to store the records in a smaller table with m entries. Using $h(k)$ function we can map a social security number in the database of employees to indexes in the table.
- **Assume:** $h(k) = k \bmod 111$
- **Then:** $h(064212848) = 064212848 \bmod 111 = 14$

$h(037149212) = 037149212 \bmod 111 = 65$

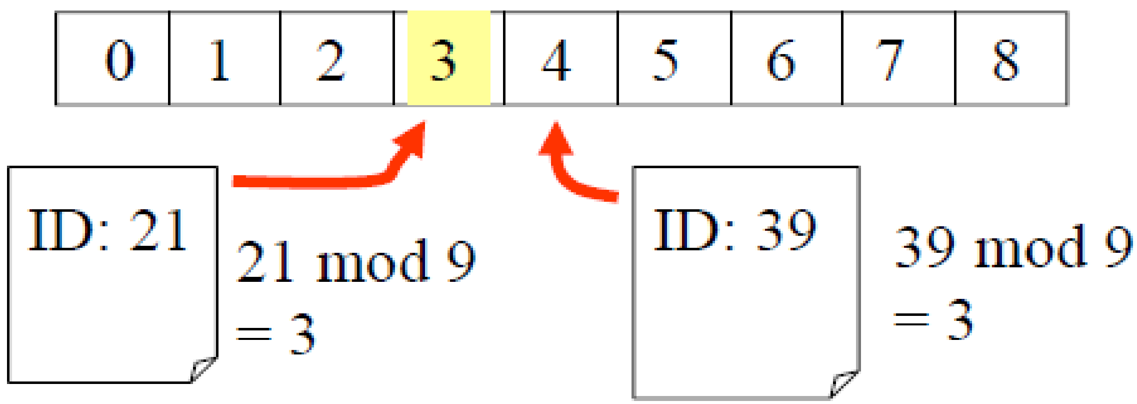
Application of Modular Arithmetic (2)

- Because a hashing function is not one-to-one (because there are more possible keys than memory locations), more than one file may be assigned to a memory location.
- When this happens, we say that a **collision** occurs.



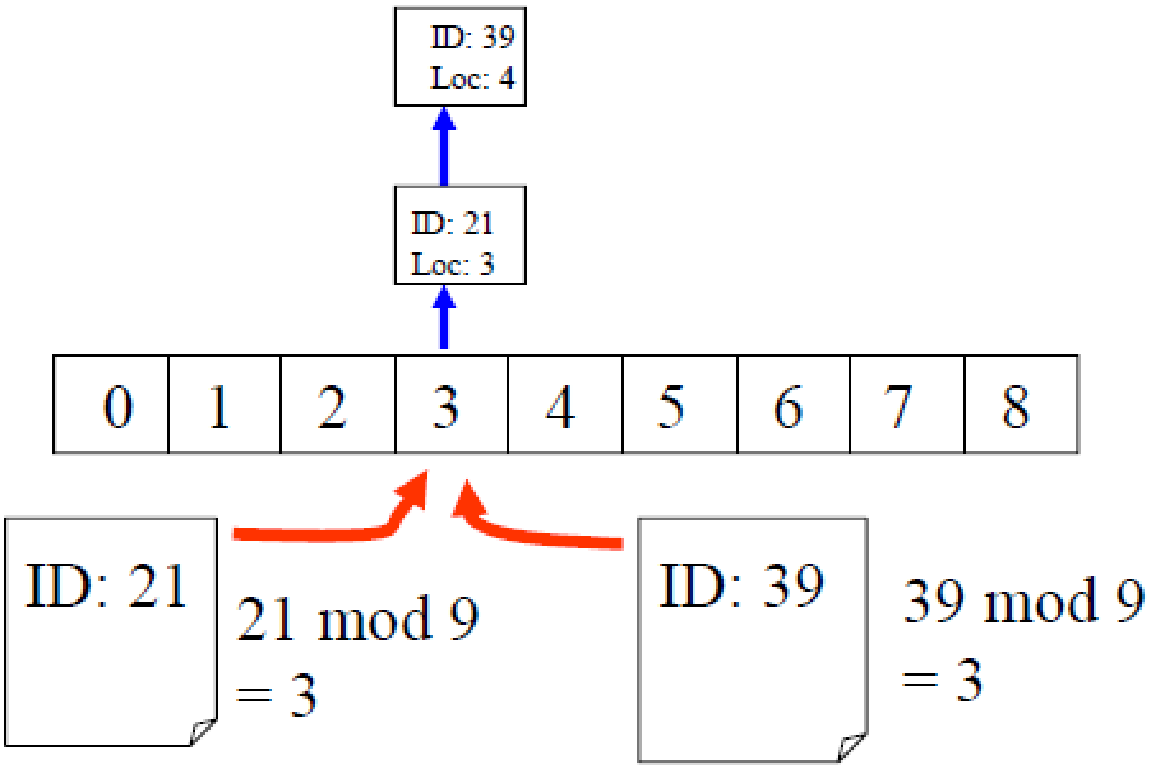
Application of Modular Arithmetic (2)

- **Solution 1**: move to the next available location
  - Method is represented by a sequence of hash functions to try:
    $h_0(k) = k \textbf{ mod } n$, $h_1(k) = (k+1) \textbf{ mod } n$, $h_2(k) = (k+2) \textbf{ mod } n, \ldots,$
    $h_m(k) = (k+m) \textbf{ mod } n$.



Application of Modular Arithmetic (2)

- **Solution 2**: remember the exact location in a secondary structure that is searched sequentially



Application of Modular Arithmetic (3)

- Cryptology: encryption of messages using **Caesar cipher**
- Shift letters in the message by 3, last three letters mapped to the first 3 letters, e.g. A is shifted to D, X is shifted to A.
- **How to represent the idea of a shift by 3?**
  - There are 26 letters in the alphabet.
  - Assign each of them a number from 0,1, 2, 3, .. 25 according to the alphabetical order.
  - The encryption of the letter with an index p is represented as:

$$f(p) = (p + 3) \bmod 26$$

  - Plaintext: **I LIKE DISCRETE MATH**
  - Ciphertext (encrypted message): **L OLNH GLYFUHVH PDVK**
- **What method would you use to decode the message:**

$$f^{-1}(p) = (p-3) \bmod 26$$