TECHNO INDIA UNIVERSITY
WESTBENGAL

Data Structures & Algorithms
(PCC-CS 301)

Dr. Debashis Das
Associate Professor
Department of CSE
Techno India University, Kolkata

Topics Covered

1. Data Searching: Hashing

Data Searching

- Searching of Data
  - Objective
    - Accessing of a particular data stored in a data structure
    - Searching process should be efficient in terms of speed
  - Type of searching
    - Sequential (or linear) Search
    - Binary Search
    - Hashing
    - Depth First Search
    - Breadth First Search

Performed on linear data structures

Performed on hash table

Performed on Graph data structures

Data Searching

- Why hashing ?

  - Time complexity
    - Linear search: O(n) *where 'n' is total number of elements*
    - Binary search: O(log n) *where 'n' is total number of elements*
    - DFS & BFS : O(V+E) *where 'V' and 'E' are number of vertices and edges*

  - Is searching possible in lesser time than the above?
    - Yes, using hashing concept
    - Time complexity: O(1)
      - *independent of the number of element stored*

Data Searching

- Hashing
  - Introduction
    - The key values are stored into **Hash Table**, one kind of data structure
    - A **hash function** is used to store keys into hash table and search them
    - Hash functions are represented as ***H(k)*** where 'k' symbolizes key values
    - The hash function finds the index, **hash value**, where the key is to be stored (or searched) in the hash table
    - It directly reaches to the key element in the hash table

Data Searching

- Hashing
  - Hash functions
    - **Division method:** hash value is estimated by dividing the key with a large number **m** (m > n , where **n** is the total number of keys), preferably prime number

$$H(k) = k(\mathrm{mod}\, m)$$

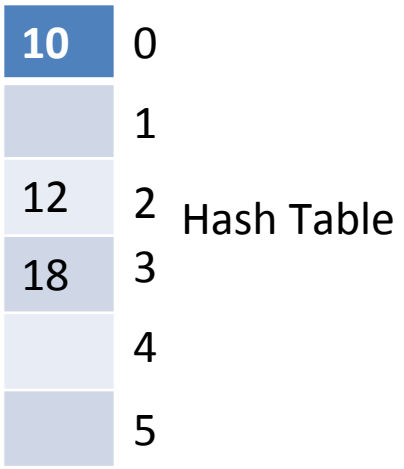
- Ex. – store {10, 18, 12} in a hash table having size 6.

m=5

$$H(10) = 10(\mathrm{mod}\, 5) = 0$$

$$H(18) = 18(\mathrm{mod}\, 5) = 3$$

$$H(12) = 12(\mathrm{mod}\, 5) = 2$$



Data Searching

- Hashing
  - Hash functions
    - **Mid-square method:** key is squared and hash value ***x*** is obtained by removing digits from both ends of k^2

$$H(k) = x$$

    - Ex. – store {3205, 7148, 2345}, hash table size 99, m=97

K:	3205	7148	2345
K^2:	10272025	51093904	5499025
H(k):	72	93	99

extra digit removes from right

Data Searching

- Hashing
  - Hash functions
    - **Folding method:** the key is partitioned into several parts, each part will contain same number of digits as the required address possibly except the last part. Hash value is obtained by adding the parts and ignoring the last carry

$$H(k) = k1 + k2 + \ldots + kr$$

    - Ex.-

K: { 3205, 7148, 2345 }

H(3205): 32 + 05 = 37
H(7148): 71 + 48 = 19
H(2345): 23 + 45 = 68

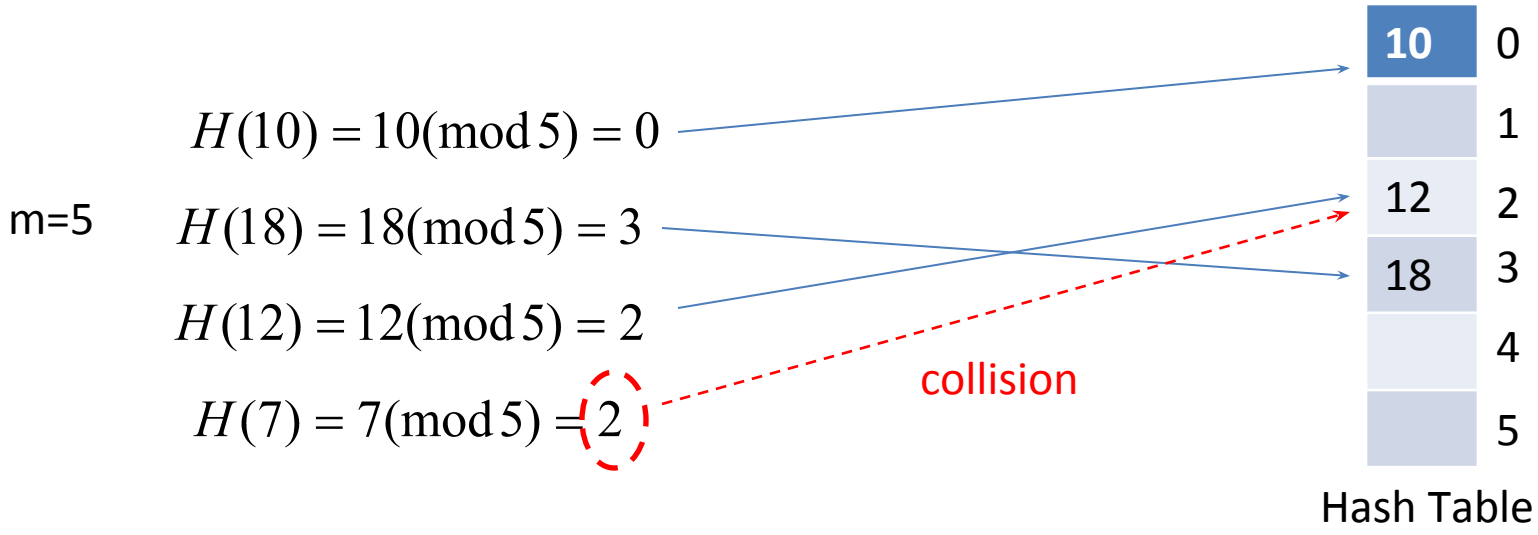
Every key is divided into 2 parts here

Data Searching

- Hashing
  - Collision resolution
    - **Open addressing**
      - Linear probing
      - Quadratic probing
      - Double hashing
    - **Chaining**

Data Searching

- Hashing
  - Collision
    - **Collision:** If more than one key results same hash value
    - Ex. – store {10, 18, 12, 7} in a hash table having size 6.



Data Searching

- Hashing
  - Collision resolution: Linear probing

    It searches for the next available space to store the key that encounters a collision (it searches circularly, after last position it comes back to the starting index)
    - Ex. - Key values = {10, 18, 12, 7}

m=5

$H(10) = 10 (\text{mod}\, 5) = 0$

$H(18) = 18 (\text{mod}\, 5) = 3$

$H(12) = 12 (\text{mod}\, 5) = 2$

$H(7) = 7 (\text{mod}\, 5) = 2$

collision

solution

10	0
	1
12	2
18	3
7	4
	5

Hash Table

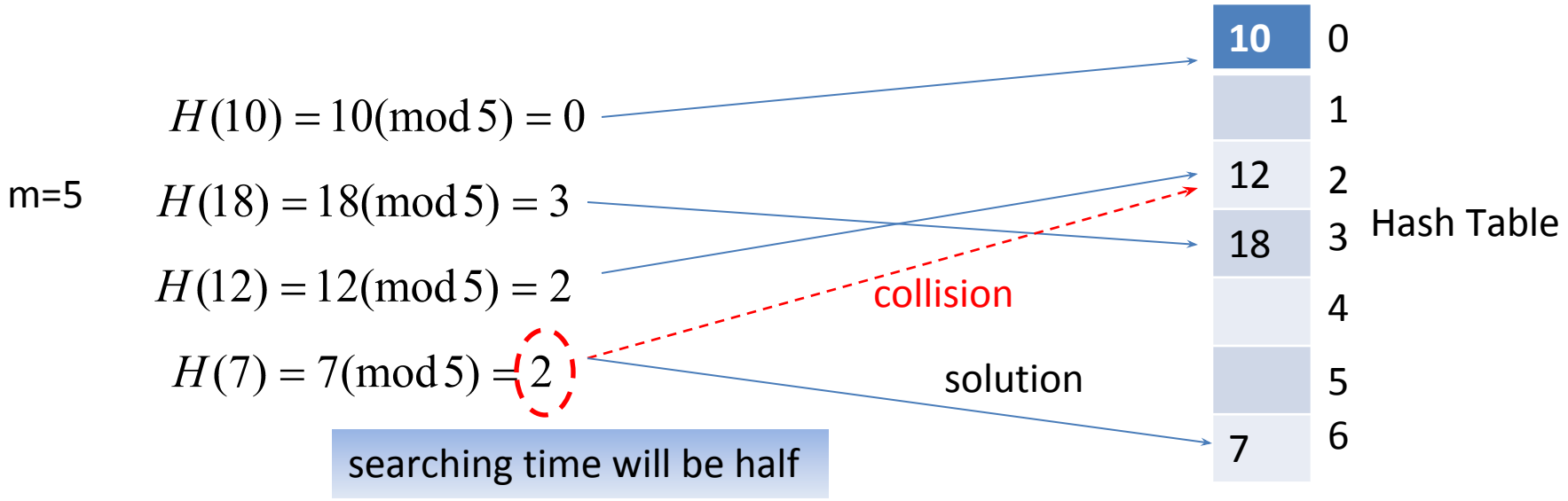
It increases the searching time

Data Searching

- Hashing
  - Collision resolution: Quadratic probing

    If hash value at collision is *h*, it searches at h, h+1, h+4, …, $h+i^2$

    - Ex. - Key values = {10, 18, 12, 7}



Data Searching

- Hashing
  - Collision resolution: Double hashing

    Two hash functions are used to break the collision, H(k)= h and H'(k)= h', searches the next index as *h, h+h', h+2h', h+3h' …*

- Ex. - Key values = {10, 18, 12, 7}

m=5

m'=3

$H(10) = 10 (\mathrm{mod}\, 5) = 0$

$H(18) = 18 (\mathrm{mod}\, 5) = 3$

$H(12) = 12 (\mathrm{mod}\, 5) = 2$

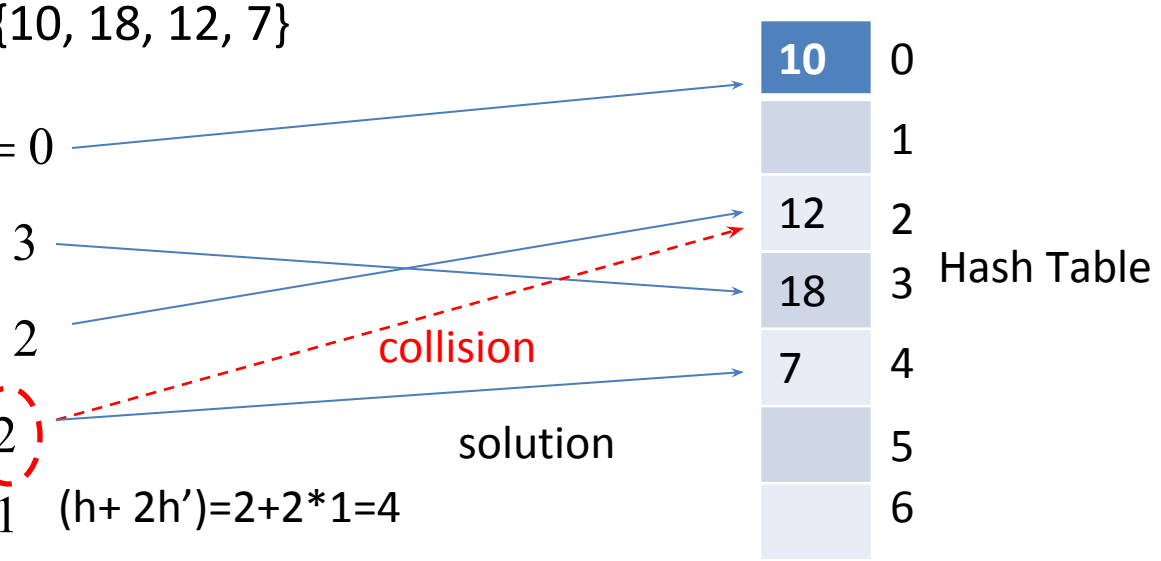
$H(7) = 7 (\mathrm{mod}\, 5) = 2$

$H'(7) = 7 (\mathrm{mod}\, 3) = 1$

collision

solution

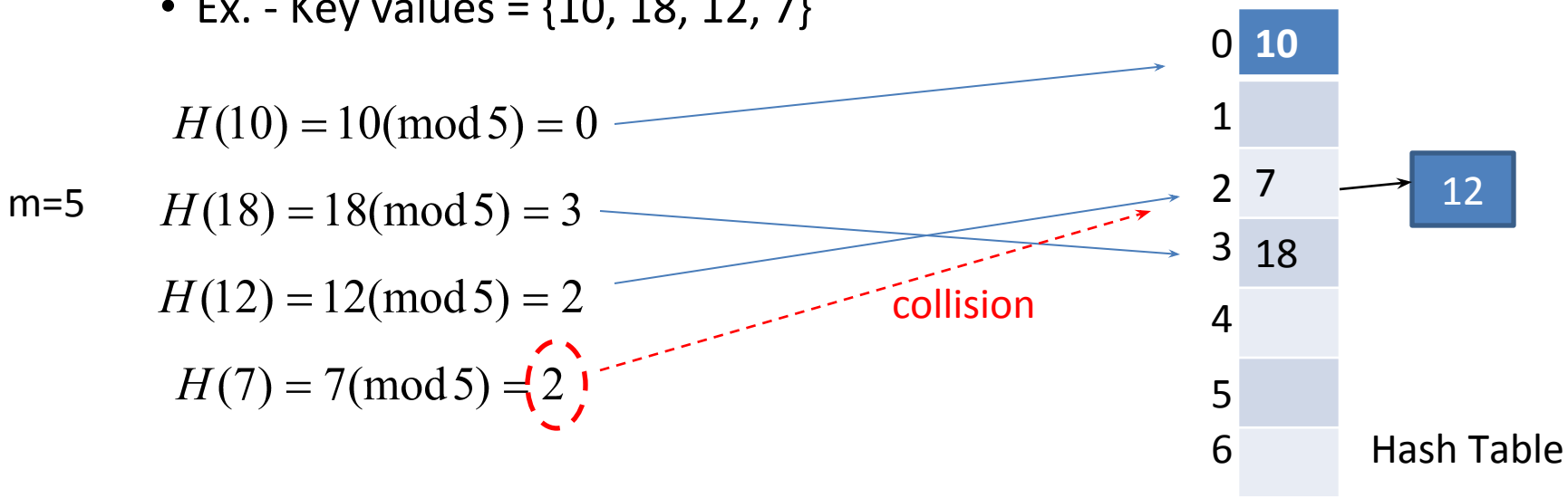
(h+ 2h')=2+2*1=4



Data Searching

- Hashing
  - Collision resolution: Chaining

    If collision occurs in some index, new key is added as linked address. New key is added, conventionally, at the beginning of the list
    - Ex. - Key values = {10, 18, 12, 7}



Data Searching

- Hashing
  - **Important notes**
    - To solve collisions, chaining method is more accepted than open addressing method
    - If stored data frequently changes (high number of deletion), open addressing schemes increase the searching
    - There is no fixed calculation to define the hash table size, it depends on the problem statement and application
    - However, to obtain efficient searching, hash table size may be considered as [no. of key * 1.3] even if there are good number of collisions. It reduces the percentage of unsuccessful search

Queries?