Parul® University

Unit 5
Hashing

Amir Hussain

Cyber Security Trainer

Computer Science & Engineering

Parul®
University

DIGITAL LEARNING CONTENT

Topic

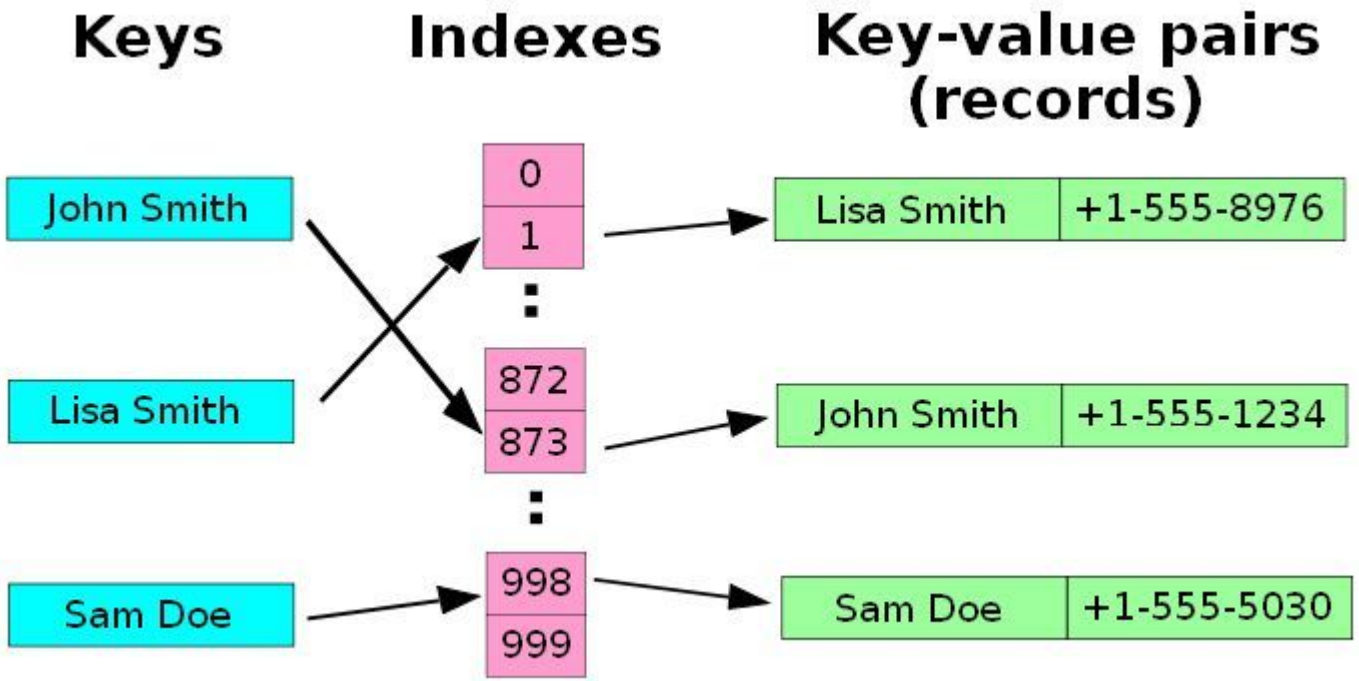
Hashing

Introduction

- In CS, a **hash table**, or a **hash map**, is a data structure that associates keys (names) with values (attributes).
  - Look-Up Table
  - Dictionary
  - Cache
  - Extended Array

Introduction

Example: A small phone book as a hash table.



Source: Wikipedia

Introduction

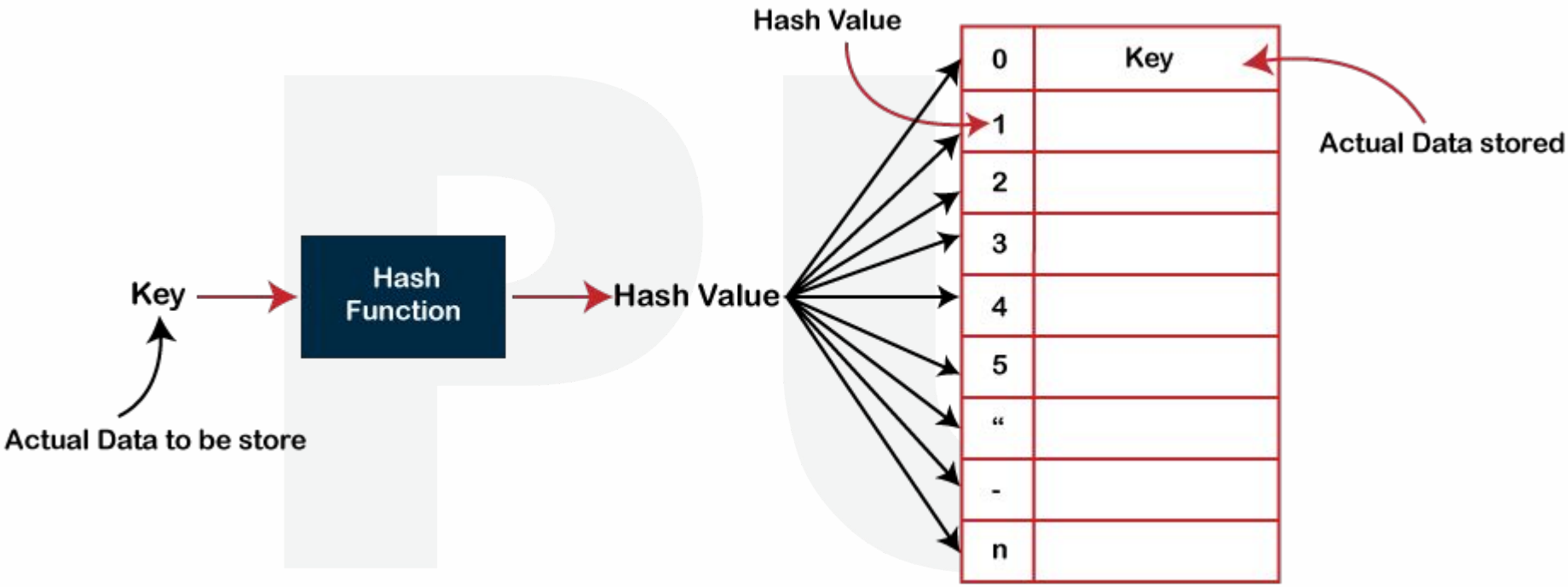
- Hashing is one of the searching techniques that uses a constant time. The time complexity in hashing is O(1).

- In Hashing technique, the hash table and hash function are used. Using the hash function, we can calculate the address at which the value can be stored.

- The main idea behind the hashing is to create the (key/value) pairs. If the key is given, then the algorithm computes the index at which the value would be stored. It can be written as:

Index = hash(key)

Introduction



Hash Function

- It is the mathematical function which takes unique key as an argument and return a unique memory location which is used to store and retrieve the data related to key.

- Ex. m = H(key)
  - Where, **m** is memory location
  - If the key is 3205 and if we are using mid-square method as hash function them m=72. Which means, we are 72th location of memory to store and retrieve data related to key 3205.

Hash Function

- **Some of the hash functions are:**
  - Mid-Square method
  - Division method
  - Folding method

Hash Function - Mid-Square

- In this method, first we square the given key, then we find out the mid of this squared key by first putting left & then right, moving to mid.

- Ex. Key = 3205
  - Step 1: (key)^2 = (3205)^2 = 10272025
  - Step 2: deleting from left then right, moving towards mid.

    therefore, m = 72

Hash Function - Division

- Using the formula m = key mod D, we find out the value of m to retrieve the required data.
- D is the largest prime no. between lower and upper limit of memory.

- Ex. Key = 3205

  m = key % D,

  suppose lower limit = 0 and upper limit 99 then D will be 97

  Therefore, m = 3205 % 97 = 04

Hash Function - Folding

- We fold the given no. into the digits which will be equal to the digits in upper limit and the we add them.
- Ex 1: key = 3205
  - Step 1: folding -> 32 and 05
  - Step 2: adding -> 37 is the required m
- Ex 2: key = 2209
  - Step 1: folding -> 22 and 09
  - Step 2: adding -> 31 is the required m

Hash Function - Folding : Collision

- **Ex 3: key = 3205**
  - Step 1: folding -> 32 and 05
  - Step 2: adding -> 37 is the required m
- **Ex 4: key = 3106**
  - Step 1: folding -> 31 and 06
  - Step 2: adding -> 37 is the required m
- The situation arises in above 2 example is known as collision. Which means for the different keys we are getting same memory address.

Collision Resolution Technique

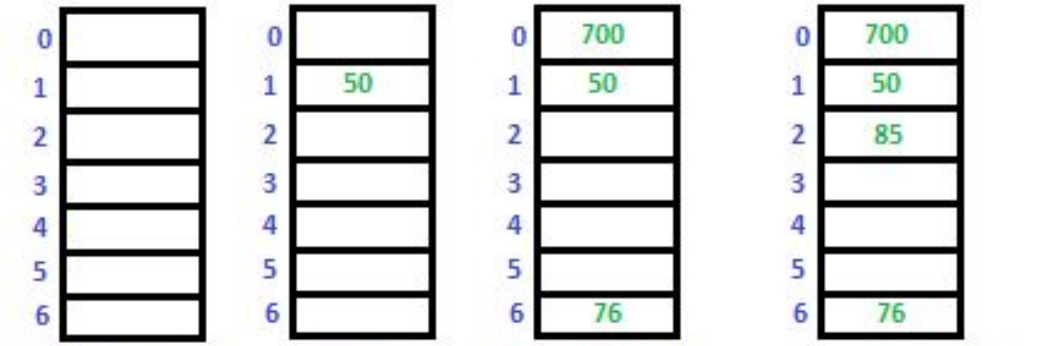
The following techniques are used to deal with the collision problem:

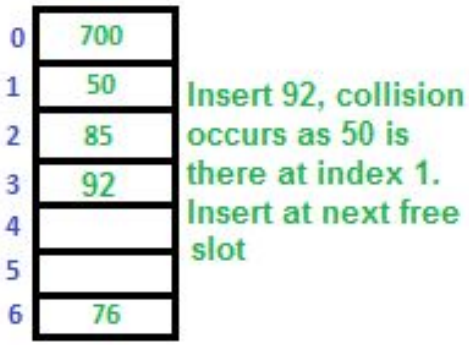
1. Probing
   a) Linear
   b) Quadratic
2. Rehashing
3. Chaining

Probing - Linear

- In linear probing, we linearly probe for next slot.
- We find next empty location as:
  - M = (p + i) % SIZE
  - Ex. Let us consider a simple hash function as “key mod 7” and sequence of keys as 50, 700, 76, 85, 92, 73, 101.

Probing - Linear





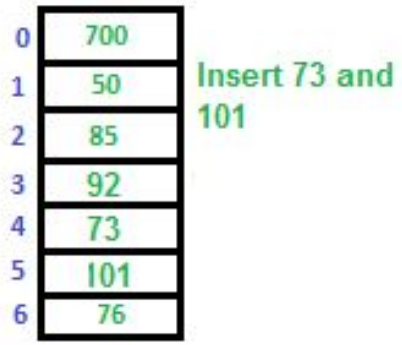


Image source: google

Probing - Quadratic

- In this method, we find next empty location as:
  - M = (p + i^2) % SIZE
- Rest process is same as linear probing.

Rehashing

- Also known as Double hashing.
- In this method to search the next empty location, we apply different hash functions in a particular order to store and retrieve colliding keys.
- Double hashing : F(i) = i * Hash2(X)

Rehashing

- Ex.
  - let hash(x) be the slot index computed using hash function.
  - If slot hash(x) % S is full, then we try (hash(x) + 1*hash2(x)) % S
  - If (hash(x) + 1*hash2(x)) % S is also full, then we try (hash(x) + 2*hash2(x)) % S
  - If (hash(x) + 2*hash2(x)) % S is also full, then we try (hash(x) + 3*hash2(x)) % S

Chaining

- Chaining is also known as Bucketing

- In this we use a separate data structure called as linked list (Chain) of colliding keys. As collision occurs we add related colliding key at the end of link list of colliding frame.

Chaining

- Ex. Let us consider a simple hash function as “key mod 7” and sequence of keys as 50, 700, 76, 85, 92, 73, 101.

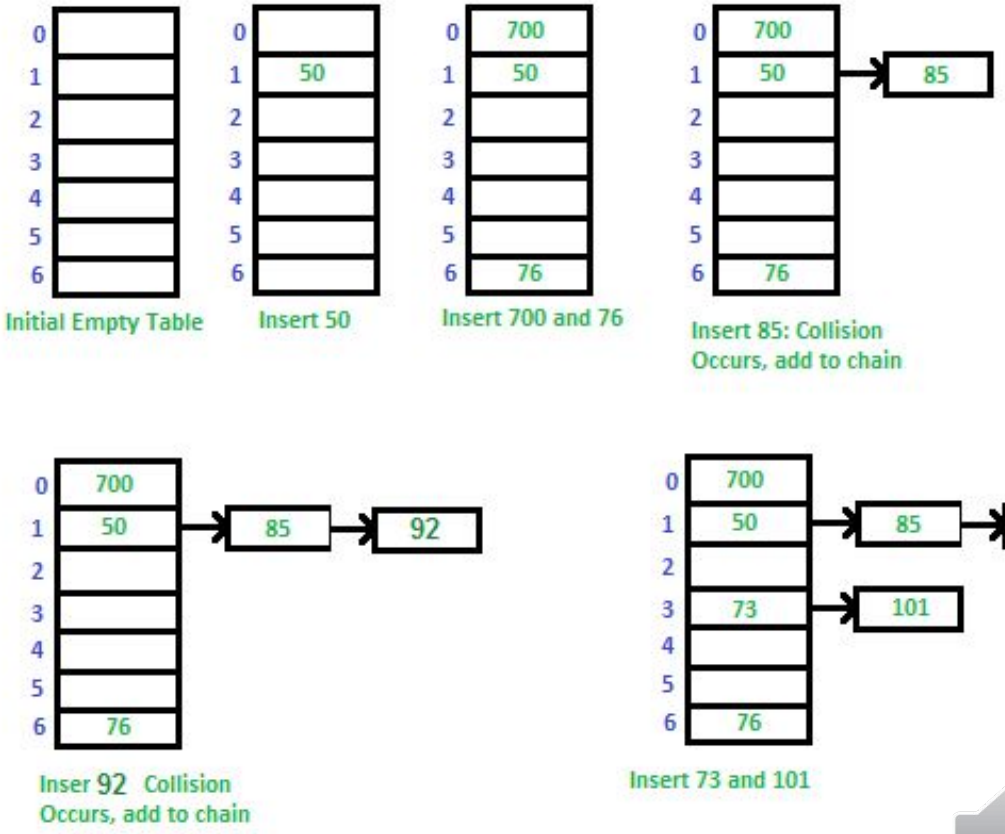


Image source: google

Static Hashing

It is a hashing technique that enables users to lookup a definite data set. Meaning, the data in the directory is not changing, it is "Static" or fixed.

In this hashing technique, the resulting number of data buckets in memory remains constant.

Operations Provided by Static Hashing

- **Delete –** Search a record address and delete a record at the same address or delete a chunk of records from records for that address in memory.

- **Insertion –** While entering a new record using static hashing, the hash function (h) calculates bucket address "h(K)" for the search key (k), where the record is going to be stored.

Image source: google

Static Hashing

- **Search** − A record can be obtained using a hash function by locating the address of the bucket where the data is stored.

- **Update** − It supports updating a record once it is traced in the data bucket

Image source: google

Dynamic Hashing

- It is a hashing technique that enables users to lookup a dynamic data set. Means, the data set is modified by adding data to or removing the data from, on demand hence the name ‘Dynamic’ hashing.

- In this hashing technique, the resulting number of data buckets in memory is ever-changing.

Image source: google

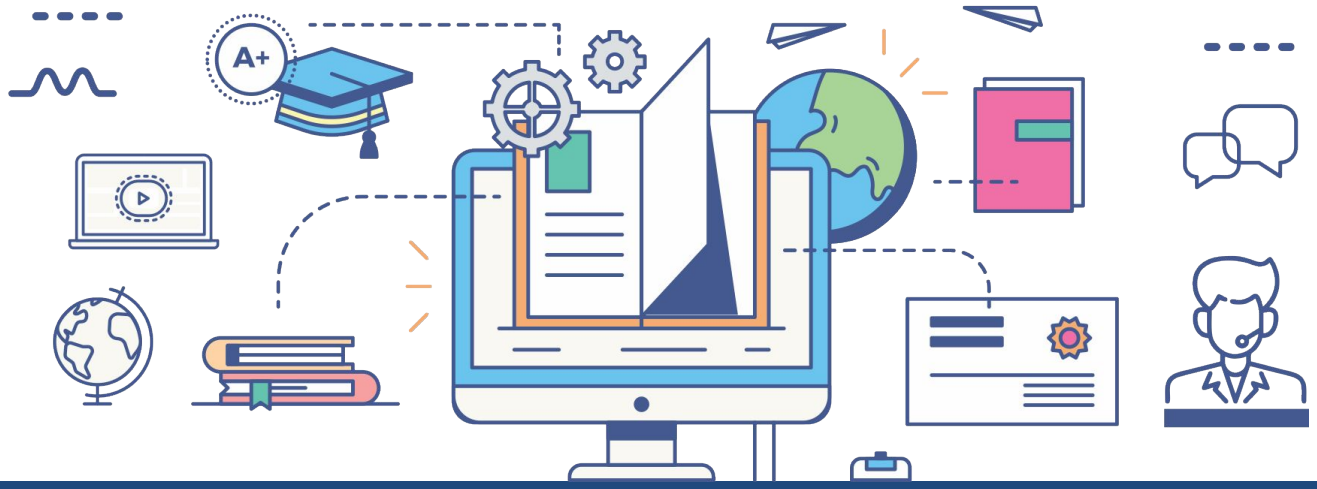
Dynamic Hashing

Operations Provided by Dynamic Hashing

- **Delete** – Locate the desired location and support deleting data (or a chunk of data) at that location.
- **Insertion** – Support inserting new data into the data bucket if there is a space available in the data bucket.
- **Query** – Perform querying to compute the bucket address.
- **Update** – Perform a query to update the data.

Image source: google

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in