# SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

**An Autonomous Institution**

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

## DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE NAME :23ITB201-DATA STRUCTURES & ALGORITHMS

II YEAR / III SEMESTER

Unit III- **SORTING, SEARCHING & HASHING**

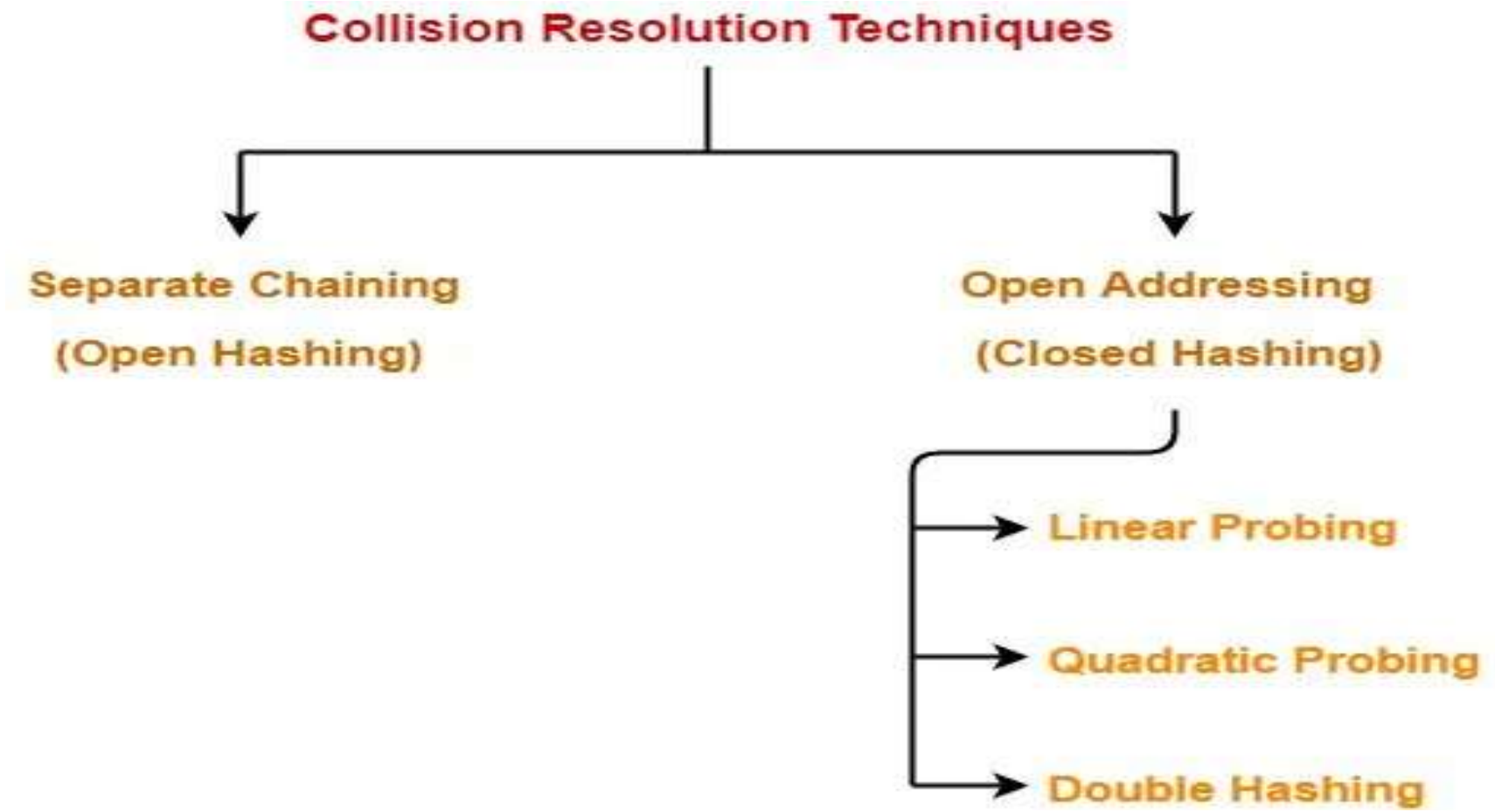
Topic :Collision Handling

A collision occurs when more than one value to be hashed by a particular hash function hash to the same slot in the table or data structure (hash table) being generated by the hash function.

When two or more keys have the same hash value, a collision happens. To handle this collision, we use collision resolution techniques.

Collision Resolution Techniques
There are two types of collision resolution techniques.

Separate chaining (open hashing)
Open addressing (closed hashing)


Collision Resolution Techniques
Separate Chaining
(Open Hashing)
Open Addressing
(Closed Hashing)
Linear Probing
Quadratic Probing
Double Hashing

# Separate chaining (open hashing)

To handle the collision,

This technique creates a linked list to the slot for which collision occurs.

The new key is then inserted in the linked list.

These linked lists to the slots appear like chains.

That is why, this technique is called as separate chaining.

# Time Complexity-

## For Searching-

In worst case, all the keys might map to the same bucket of the hash table.

In such a case, all the keys will be present in a single linked list.

Sequential search will have to be performed on the linked list to perform the search.

So, time taken for searching in worst case is O(n).

## For Deletion-

In worst case, the key might have to be searched first and then deleted.

In worst case, time taken for searching is O(n).

So, time taken for deletion in worst case is O(n).

## Load Factor (α)-

Load factor (α) is defined as-

$$\text{Load Factor } (\alpha) = \frac{\text{Number of elements present in the hash table}}{\text{Total size of the hash table}}$$

If Load factor (α) = constant, then time complexity of Insert, Search, Delete = Θ(1)

**PRACTICE PROBLEM BASED ON SEPARATE CHAINING-**

**Problem-**

Using the hash function 'key mod 7', insert the following sequence of keys in the hash table-

50, 700, 76, 85, 92, 73 and 101

## Step-01:

Draw an empty hash table.

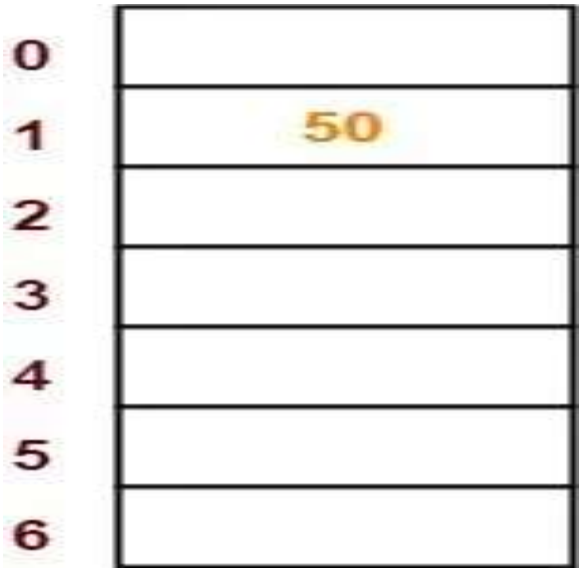
For the given hash function, the possible range of hash values is [0, 6].

So, draw an empty hash table consisting of 7 buckets as-

## Step-02:

Insert the given keys in the hash table one by one.

The first key to be inserted in the hash table = 50.
Bucket of the hash table to which key 50 maps = 50 mod 7 = 1.
So, key 50 will be inserted in bucket-1 of the hash table as-



## Step-03:

The next key to be inserted in the hash table = 700.
Bucket of the hash table to which key 700 maps = 700 mod 7 = 0.
So, key 700 will be inserted in bucket-0 of the hash table as-

| | |
|---|---|
| 0 | 700 |
| 1 | 50 |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |

## Step-04:

The next key to be inserted in the hash table = 76.
Bucket of the hash table to which key 76 maps = 76 mod 7 = 6.
So, key 76 will be inserted in bucket-6 of the hash table as-

| | |
|---|---|
| 0 | 700 |
| 1 | 50 |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | 76 |

**Step-05:**

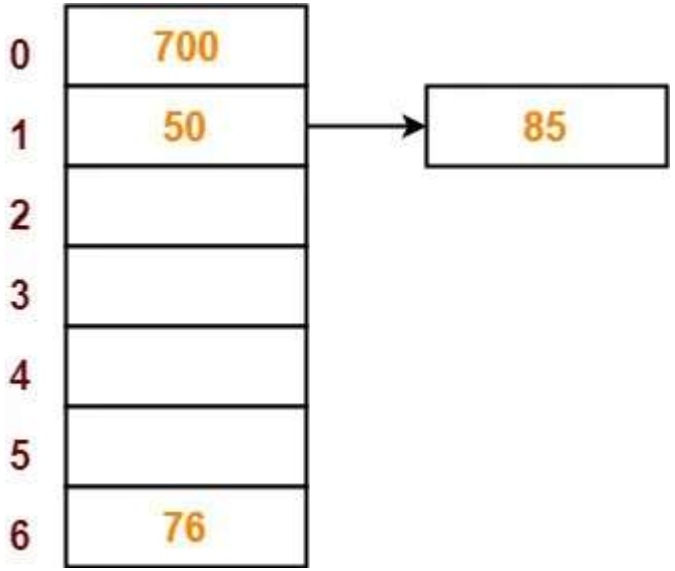
The next key to be inserted in the hash table = 85.

Bucket of the hash table to which key 85 maps = 85 mod 7 = 1.

Since bucket-1 is already occupied, so collision occurs.

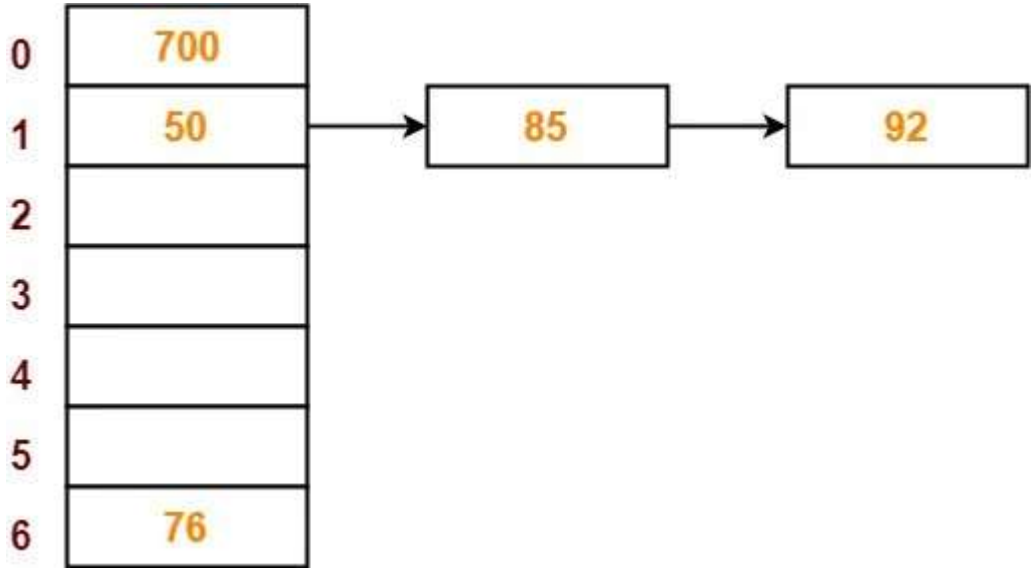
Separate chaining handles the collision by creating a linked list to bucket-1.

So, key 85 will be inserted in bucket-1 of the hash table as-



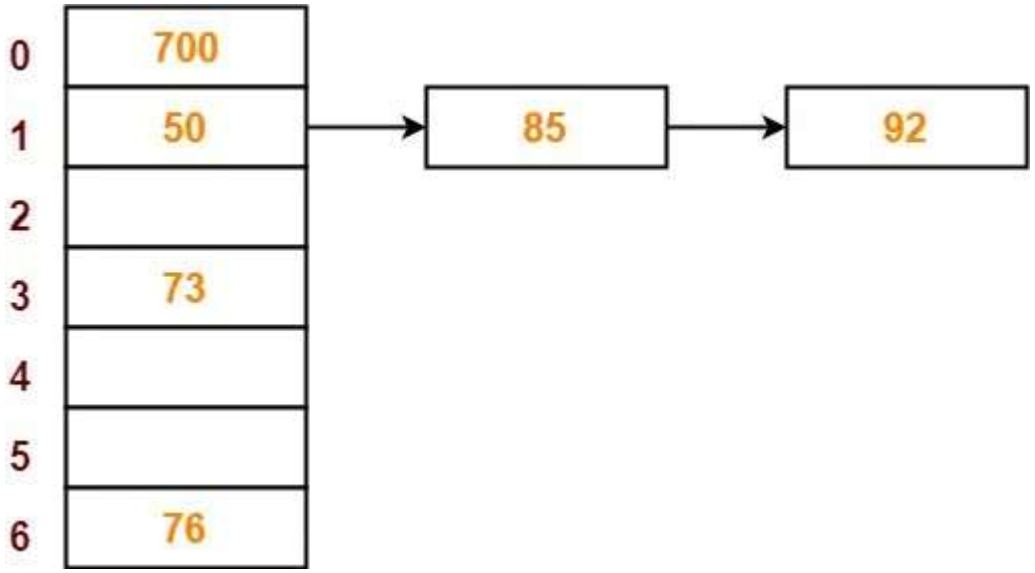
## Step-06:

The next key to be inserted in the hash table = 92.
Bucket of the hash table to which key 92 maps = 92 mod 7 = 1.
Since bucket-1 is already occupied, so collision occurs.
Separate chaining handles the collision by creating a linked list to bucket-1.
So, key 92 will be inserted in bucket-1 of the hash table as-



## Step-07:

The next key to be inserted in the hash table = 73.
Bucket of the hash table to which key 73 maps = 73 mod 7 = 3.
So, key 73 will be inserted in bucket-3 of the hash table as-



**Step-08:**

The next key to be inserted in the hash table = 101.
Bucket of the hash table to which key 101 maps = 101 mod 7 = 3.
Since bucket-3 is already occupied, so collision occurs.
Separate chaining handles the collision by creating a linked list to bucket-3.
So, key 101 will be inserted in bucket-3 of the hash table as-

???????

## Open Addressing for Collision Handling

Similar to separate chaining, open addressing is a technique for dealing with collisions.

In Open Addressing, the hash table alone houses all of the elements.

The size of the table must therefore always be more than or equal to the total number of keys at all times (Note that we can increase table size by copying old data if needed).

This strategy is often referred to as closed hashing. The foundation of this entire process is probing.

## Open Addressing
## Open addressing is when

All the keys are kept inside the hash table, unlike separate chaining.
The hash table contains the only key information.
The methods for open addressing are as follows:

Linear Probing
Quadratic Probing
Double Hashing

## (a) Linear probing

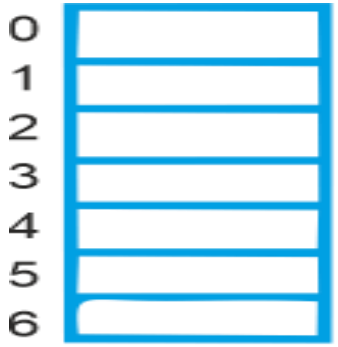
In linear probing, the hash table is systematically examined beginning at the hash's initial point. If the site we receive is already occupied, we look for a different one.

The rehashing function is as follows: table-size = (n+1)% rehash(key). As may be seen in the sample below, the usual space between two probes is 1.

Let S be the size of the table and let hash(x) be the slot index calculated using a hash algorithm.

If slot hash (x) % S is full, then we try ( hash (x) + 1 ) % S
If ( hash (x) + 1 ) % S is also full, then we try ( hash (x) + 2) % S
If ( hash (x) + 2 ) % S is also full, then we try ( hash (x) + 3 ) % S

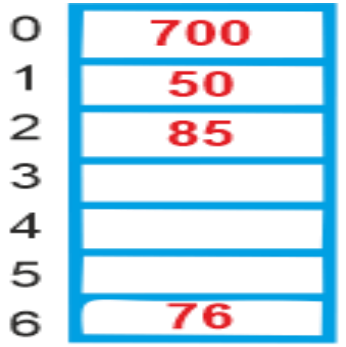
Let's use "key mod 7" as a simple hash function with the following keys: 50, 700, 76, 85, 92, 73, 101



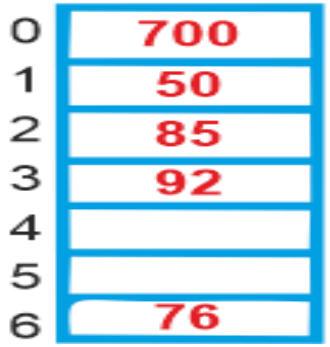

Initial Empty Table


Insert 50

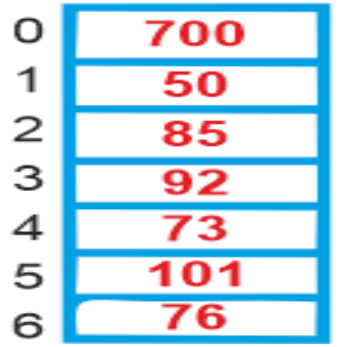

Insert 700 and 76




Insert 85: Collision Occurs, insert 85 at next free slot.




Insert 92: Collision Occurs as 50 is there at index 1. Insert at nest free slot




Insert 73 and 101

## (b) Quadratic probing

If you pay close attention, you will notice that the hash value will cause the interval between probes to grow.

The above-discussed clustering issue can be resolved with the aid of the quadratic probing technique. The mid-square method is another name for this approach.

We search for the i2'th slot in the i'th iteration using this strategy. We always begin where the hash was generated. We check the other slots if only the location is taken.

let hash (x) be the slot index computed using hash function.

If slot hash(x) % S is full, then we try ( hash (x) + 1*1 ) % S

If ( hash (x) + 1*1 ) % S is also full, then we try ( hash (x) + 2*2 ) % S

If ( hash (x) + 2*2 ) % S is also full, then we try ( hash (x) + 3*3 ) % S

## c) Double Hash

Another hash function calculates the gaps that exist between the probes. Clustering is optimally reduced by the use of double hashing.

This method uses a different hash function to generate the increments for the probing sequence. We search for the slot i*hash2(x) in the i'th rotation using another hash algorithm, hash2(x).

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

| S. No. | Separate Chaining | Open Addressing |
|---|---|---|
| 1. | Chaining is easier to put into practise. | Open Addressing calls for increased processing power. |
| 2. | Hash tables never run out of space when chaining since we can always add new elements. | Table may fill up when addressing in open fashion. |
| 3. | Chaining is less susceptible to load or the hash function. | To prevent clustering and load factor, open addressing calls for extra caution. |

| | | |
|---|---|---|
| **4.** | When it is unclear how many or how frequently keys might be added or removed, chaining is typically utilised. | When the frequency and quantity of keys are known, open addressing is employed. |
| **5.** | Chaining's cache performance is poor since keys are stored in linked lists. | Since everything is stored in the same table, open addressing improves cache speed. |
| **6.** | Space wastage (Some Parts of hash table in chaining are never used). | A slot can be used in open addressing even if an input doesn't map to it. |

## MCQ1

**Which of the following is a common collision handling technique used in hash tables?**

A) Linear search

B) Linear probing

C) Quick sort

D) Binary search

*Answer*: **B) Linear probing**

**MCQ 2:**

**In the collision handling method known as "chaining," how are collisions resolved?**

A) By storing all collided elements in an array

B) By moving the collided element to the next available index

C) By creating a linked list at each hash table index

D) By rehashing collided elements

*Correct Answer*: **C) By creating a linked list at each hash table index**

**MCQ 3:**

**Which of the following statements about open addressing is true?**

A) It stores collided elements in a secondary hash table

B) It involves storing collided elements in the same hash table using a sequence of probes

C) It always leads to the same hash index for collided elements

D) It uses external storage to handle collisions

*Correct Answer*: **B) It involves storing collided elements in the same hash table using a sequence of probes**

Thank you……