

* Hashing

- Hashing is the process of mapping large amount of data item to smaller table with the help of hashing function.
- Hashing is also known as Hashing Algorithm.
- Hashing is a technique or process of mapping keys, values into the hash table by using a hash function.

hash function A fixed process converts a key to a hash key is known as hash function.

- This function takes a key and maps it to a value of a certain length which is called a hash value.

Hash Table Hash table is a data structure used to store key-value pairs.

- It is a collection of items stored to make it easy to find them later.
- It contains value based on the key

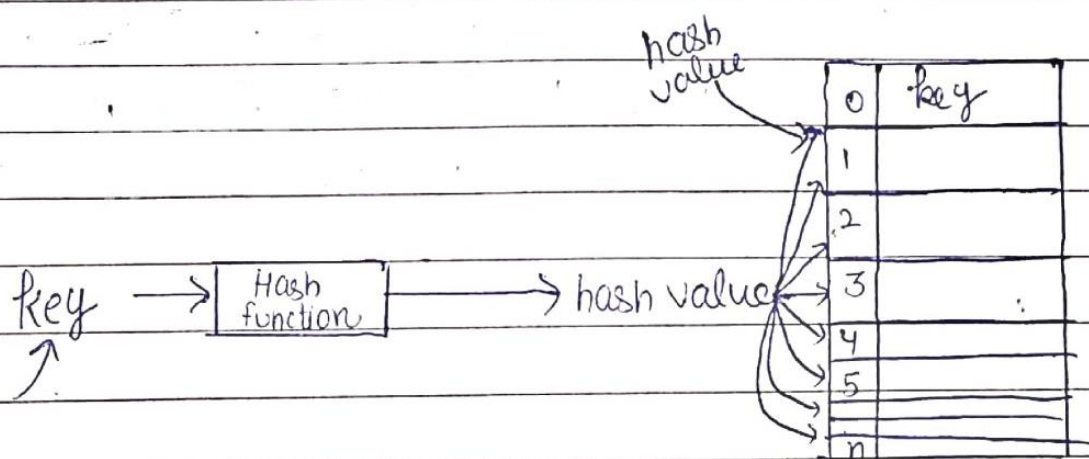
0	1	2	3	4	5	6	7	8

Fig hash table

* Hash function :-

- Hash function is a function which is applied on a key by which it produces an integer, which can be used as an address of hash Table.
- The integer returned by the hash function is called hash key.

$$\text{index} = \text{hash}(\text{key})$$



Types of Hash function :-

There are the three ways of calculating the hash function :-

- Division Method
- Folding Method
- Mid Square Method

1. Division Method :-

One common method of determining a hash key is the division method of hashing.

formula :- $key\ value \% m$

m is the size of table

It takes an item and divides it by the table size and returns the remainder as its hash value.

for example :- {26, 70, 18, 31, 54, 93}

Data Item	$k \% m$	Hash value	Hash Table	
26	$26 \% 10 = 6$	6	0	70
70	$70 \% 10 = 0$	0	1	31
18	$18 \% 10 = 8$	8	2	
31	$31 \% 10 = 1$	1	3	93
54	$54 \% 10 = 4$	4	4	54
93	$93 \% 10 = 3$	3	5	
			6	26
			7	
			8	18
			9	

② Folding Method :- In this, the key is divided into separate parts and by using some simple operations these parts are

Combined to produce a hash key.

(I) method) Divide the key k into 2 parts and adding parts the following hash Tables

$$H(3205) = 32 + 05 = 37$$
$$H(2345) = 23 + 45 = 68$$

(II) Method

Here we dealing with a hash Table with index from 00 to 99 i.e. two digit hash table. So we divide k numbers of two digits

k	2103
$k_1 k_2 k_3$	21, 03
$H(k)$	$H(2103)$
$= (k_1 + k_2 + k_3)$	$21 + 03 = 24$

Example (II)

Consider a record of 12465512. So it will divided into parts 124 + 655 + 12. After dividing the parts combine these parts by adding it.

$$H(\text{key}) = 124 + 655 + 12$$
$$= 791$$

$$H(123) = [1 + 2 + 3 = 6]$$

$$H(43) = [4 + 3 = 7]$$

$$H[56] = [5 + 6 = 11]$$

0	
1	56
2	
3	
4	
5	
6	123
7	43

③ Mid Square Method:-

- The mid square method is a very good hashing method.
- In this method, first key is squared and then mid part of the result is taken as the index.
- For example:- if we want to place a record of 3101 and size of table is 1000.

So:- $3101 * 3101 = 9616201$
 i.e. $h(3101) = 162$ (middle 3 digit)

K	4147	3750	2103
K ²	17197609	14062500	4432609
h(K)	97	62	22

Removing digits from square of key values

***978**
 ***625**
 22

hash address	keys
0	
1	
--	
--	
22	3103
--	
62	3750
--	
--	
97	4147
--	
99	

Hash Table with Mid Sq Division

3

Collision :-

- o A collision occurs when a hashing algorithm produces an address for a key and that address is already occupied.
- o When the two different values have the same value, then the problem occurs between the two values, known as a collision.

Example: $\rightarrow k(101) = 101 \text{ mod } 10 = 1$

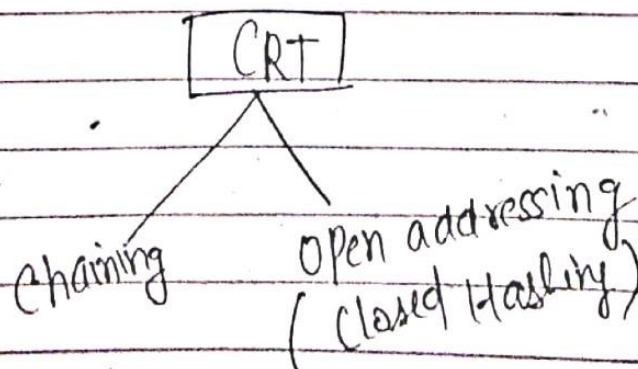
$k_2(141) = 141 \text{ mod } 10 = 1$

Collision occurred.

Collision Resolution Technique (CRT) :-

The following are the collision techniques:-

- (i) Open Addressing
- (ii) Separate Chaining



(ii) Open Addressing:-

- In open addressing method, when a key is colliding with another key, the collision is resolved by finding a nearest empty space by probing the cells.
- Home Area address are searched for an open or unoccupied element where no data can be placed.

Types of Open Addressing:-

1. Linear Probing
2. Quadratic Probing
3. Double Hashing

1. Linear Probing:-

- It is kind of Open Addressing.
- It is very easy and simple method to resolve or to handle the collision.
- In this collision can be solved by placing the second record linearly down, whenever the empty place is found.

- The Position in which a key can be stored is found by sequentially searching all positions starting from the position calculated by hash function until an empty cell is found. This type of probing is called Linear Probing.

Example:-

0	NULL
1	71
2	NULL
3	NULL
4	64
5	NULL
6	56
7	36
8	NULL
9	NULL

Keys = 56, 64, 36, 71

In this diagram we can see that 56 and 36 need to be placed at same bucket but by linear probing technique the records linearly placed downwards if place is empty. it can be seen 36 is placed at index 7.

② Quadratic Probing:-

Quadratic Probing is an open addressing technique that uses quadratic polynomial for searching until a empty slot is found.

Hash function =

$$H(\text{key}) = (H(\text{key}) + i^2) \cdot m$$

↑
table size

Example :- 89, 18, 49, 58, 69

$$m = 10$$

1. Hash(89) = $89 / 10 = 9$

2. Hash(18) = $18 / 10 = 8$

3. Hash(49) = $49 / 10 = 9 \rightarrow$ Collision occurs for first time $i = 1$

$$h_1(49) = (9 + 1^2) \cdot 10 = 10 \cdot 10 = \underline{0}$$

4. Hash(58) = $58 / 10 = 8 \rightarrow$ Collision occurs for first time $i = 1$

$$h_1(58) = (8 + 1^2) \cdot 10 = 9 \rightarrow$$
 Collision occurs for second time $i = 2$

$$h_2(58) = (8 + 2^2) \cdot 10 = \underline{2}$$

5. Hash(69) = $69 / 10 = 9 \rightarrow$ Collision occurs for first time $i = 1$

$$h_1(69) = (9 + 1^2) \cdot 10 = 9 \rightarrow$$
 Collision occurs for first time $i = 2$

$$h_2(69) = (9 + 2^2) \cdot 10 = \underline{3}$$

0	49
1	
2	58
3	69
4	
5	
6	
7	
8	18
9	29

//_

③ Double Hashing:-

- Double Hashing is an open addressing technique which is used to avoid the collisions
- It is a collision resolution technique which uses two hash function to handle collision.

Advantages of double hashing:-

- The technique does not yield any clusters.
- It can be the best form of probing because it can find ^{next} free slot in hash table more quickly than linear probing.

Example

$$h_1(k) = k \text{ mod } m \text{ (first hash fun.)}$$

$$h_2(k) = 8 - (k \text{ mod } 8) \text{ (Second hash fun.)}$$

Example - Suppose, we have a hash table of size 11. We want to insert keys 20, 30, 45, 70, 56 in the hash table.

$$h_1(k) = k \text{ mod } 11 \text{ (first hash function)}$$

$$h_2(k) = 8 - (k \text{ mod } 8) \text{ (Second hash fun.)}$$

	keys	Hash function	Index	Description
1.	20	$h_1(20) = 20 \text{ mod } 11$	9	No collision Occurs.
2.	34	$h_1(34) = 34 \text{ mod } 11$	1	No collision occurs.
3.	45	$h_1(45) = 45 \text{ mod } 11$	1	Collision occur because index 1 is already occupied by 34.
		$h_2(45) = 8 - (45 \text{ mod } 8) = 3$		(Now we will use the second hash function
		$h(45, 1) = (1 + 3) \text{ mod } 11$	4	to calculate the index for the key 45.
		\uparrow first collision occurs		
4.	70	$h_1(70) = 70 \text{ mod } 11$	4	collision occurs because index 4 is already occupied. Now we use the second hash function
		$h_2(70) = 8 - (70 \text{ mod } 8) = 2$	6	$i = 1$, because first collision occurs
		$h(70, 1) = (4 + 1 * 2) \text{ mod } 11$		

5-	56	$h_1(56) = 56 \text{ mod } 11$	56	Collision Occurs because index 1 is already occupied. Now we use the second hash function.
		$h_2(56) = 8 - (56 \text{ mod } 8)$ $= 8$	9	Again collision occur.
		$h(56, 1) = (1 + 1 * 8) \text{ mod } 11$		The index 9 is already occupied.
		$h(56, 2) = (1 + 2 * 8) \text{ mod } 11$	6	(value of i is incremented by 1) (Again collision occurs)
		$h(56, 3) = (1 + 3 * 8) \text{ mod } 11$ value of i is incremented	3	So, now we will store 56 at index 3.

Hash Table :-

0	
1	34
2	
3	56
4	45
5	
6	70
7	
8	
9	20
10	
11	

② Chaining :-

- also known as separate chaining.
- In this, a linked list is created from the slot in which collision has occurred, after which the new key is inserted into the linked list.
- This linked list of slots look like a chain, so it is called separate chaining.

Example- let us consider a simple hash function "key mod 7" and keys are 50, 700, 76, 85, 92, 73, 101.

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

initial empty Table Insert 50 insert 700 and 76 insert 85 collision occurs, add to chain

0	700	
1	50	→ [85] → [92]
2		
3		
4		
5		
6	76	

insert [92] Collision occur, add to chain

0	700	
1	50	→ [85] → [92]
2		
3	73	→ [101]
4		
5		
6	76	

insert 73 and 101

Advantages:-

- Simple to implement.
- Hash Table never fills up, we can always add more elements to the chain.

Disadvantages:- → If the chain becomes long then search can become worst.

- Wastage of space.