Hashing, Hash table, Hashing Techniques

Why Hashing:

- Internet has grown to millions of users and terabytes of data every data.
- It is impossible to find anything in the internet, unless we develop a new data structure to store and access the data very quickly.
- The amount of time required to look up an element in an array or linked list is either O(logn) or O(n) based on the list is sorted or not.
- New data structure called Hashing used to store and retrieve any entry with constant time O(1).
- This technique is irrelevant to size of the list and order.
- To increase the search efficiency the items to be stored in such a way as to make it easy to find them later.

Hash Table

- Hash table is a data structure to store and retrieve data very fast.
- Hash table consist of key and its value.
- Each location in the hash table is called cell or bucket.
- Hash table is implemented using array.
- An element is accessed very fast if we know the key or its index.

Example: To store the Student record in hash table, Student rollno is used as a key

Following are the basic primary operations of a hash table.

Search – Searches an element in a hash table.

Insert – inserts an element in a hash table.

Delete – Deletes an element from a hash table

Hashing

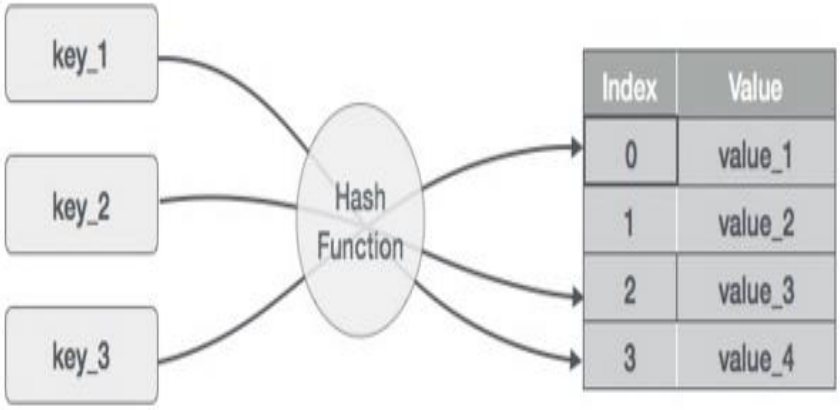
- Hashing is a technique used to generate key where an element is to be inserted or to be located from.

Note: It minimizes the number of comparisons while performing the search.

Hash function:

- To map the key value into its corresponding index in hash table hash function is used.
- A hash function h transforms a key into an index in a hash table T[0 … m-1]

  Where m is size of hash table.



- Use hash function to compute the index in the hash table for the given key value.
- Hash function returns integer value which give the index value in the hash table.

Characteristics of Hash Function

The characteristics / properties of a good hash function are

- It is efficiently computable.
- It minimizes the number of collisions.
- It distributes the keys uniformly over the table.

Types of Hash Functions :

There are various types of hash functions available such as

1. Division method
2. Mid square method

1 Division Method: The hash function depends upon the remainder of division. Typically the divisor is table length (M).

h(key)= record%M

For eg; If the record 54, 72, 89, 37 is placed in the hash table and if the table size is 10 then

54%10 = 4
72%10= 2
89 %10=9
37%10=7

0	
1	
2	72
3	
4	54
5	
6	
7	37
8	
9	89

Consider the following elements to be placed in the hash table of size 10,
37,90,45,22,17,49,55

H_1(37)=37%10=7
H_1(90)=90%10=0
H_1(45)=45%10=5
H_1(22)=22%10=2
H_1(17)=17%10=7
H_1(49)=49%10=9
H_1(55)=55%10=5

0	90
1	
2	22
3	
4	
5	45
6	
7	37
8	
9	49

In above example 17 and 55 are hashed to same location this condition is called **collision.**

Note: 1) If the hash function returns same hash key for more than one record, then this situation is called 'collision' and need to resolve it.

2 Mid Square:

In the mid square method, the key is squared and the middle or mid part of the result is used as the index. If the key is a string, it has to be preprocessed to produce a number. Consider that if we want to place a record 3111 then

3111^2 = 9678321

for the hash table of size 1000

H(3111) = 783 (the middle 3 digits)

Collision resolution techniques

There are several collision resolution techniques

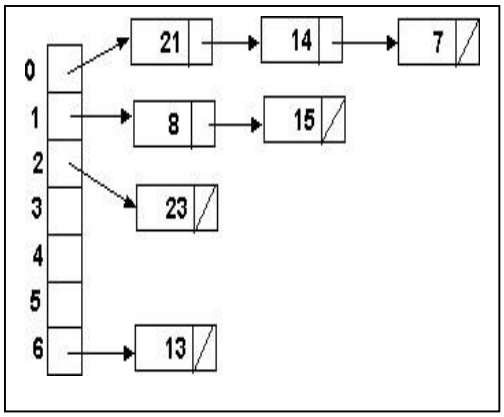
1) **Separate chaining:** an array of linked list representation

All keys that map to the same hash value are kept in a list (or "bucket").

- Hash table is implemented as array of linked list.
- All the records which are mapped to same hash address are lined together to form a linked list.

Example: Load the keys **23, 13, 21, 14, 7, 8, and 15** , in this order, in a hash table of size **7** using separate chaining with the hash function: **h(key) = key % 7**

h(23) = 23 % 7 = 2
h(13) = 13 % 7 = 6
h(21) = 21 % 7 = 0
h(14) = 14 % 7 = 0 collision
h(7) = 7 % 7 = 0 collision
h(8) = 8 % 7 = 1
h(15) = 15 % 7 = 1 collision



2) Open addressing: array based implementation

(i) Linear probing (linear search)
(ii) Quadratic probing (nonlinear search)
(iii) Double hashing (uses two hash functions