Hashing

sec 4

- Hashing is a well-known technique to search any particular element among several elements.
- It minimizes the number of comparisons while performing the search.
- 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 or Message Digest Function.
- It is a technique to convert a range of key values into a range of indexes of an array.
- It is used to facilitate the next level searching method when compared with the linear or binary search.
- Hashing allows to update and retrieve any data entry in a constant time O(1).

- An array data structure called as Hash table is used to store the data items.
- Based on the hash key value, data items are inserted into the hash table.

Hash Value

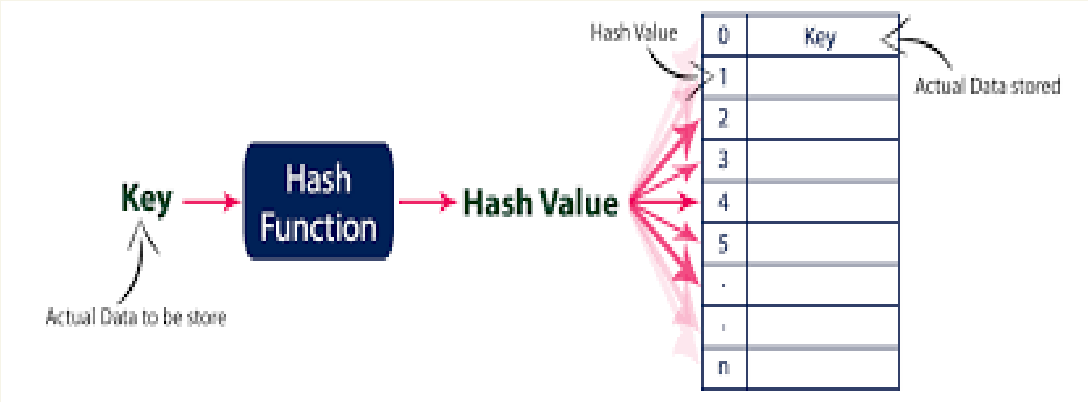
- Hash key value is a special value that serves as an index for a data item.
- It indicates where the data item should be be stored in the hash table.
- Hash key value is generated using a hash function

Hash Function

- Hash function takes the data item as an input and returns a small integer value as an output.
- The small integer value is called as a hash value.
- Hash value of the data item is then used as an index for storing it into the hash table.

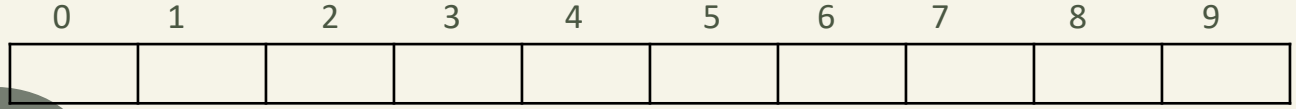
Hash table

- Hash table or hash map 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 uses a hash function to compute an index into an array of buckets or slots from which the desired value can be found.
- It is an array of list where each list is known as bucket.
- It contains value based on the key.



- Hash table with the size of n = 10. Each position of the hash table is called as Slot.
- In the next hash table, there are n slots in the table, names = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Slot 0, slot 1, slot 2 and so on. Hash table contains no items, so every slot is empty.
- As we know the mapping between an item and the slot where item belongs in the hash table is called the hash function. The hash function takes any item in the collection and returns an integer in the range of slot names between 0 to n-1.
- Suppose we have integer items {26, 70, 18, 31, 54, 93}. One common method of determining a hash key is the division method of hashing and the formula is

Hash Value = Key Value % Number of Slots in the Table



Hash Value	Value % No. of Slots	Data Item
6	26%10=6	26
0	70%10=0	70
8	18%10=8	18
1	31%10=1	31
4	54%10=4	54
3	93%10=3	93

0	1	2	3	4	5	6	7	8	9
70	31		93	54		26		18	

Collision in Hashing

- **A hashing collision occurs when two different inputs produce the same hash value.**
- **This can happen for various reasons, such as using a weak hashing algorithm, having a small hash space, or having a large number of inputs.**

- **Collision Resolution Techniques are the techniques used for resolving or handling the collision**

closed Addressing , open hashing

Chaining
buckets

Open Addressing ,closed hashing

Linear Probing
Quadratic
Double Hashing
increase size hash table

Linear Probing

How to deal with collisions?

If **h(key)** is already full,
try **(h(key) + 1) % TableSize**. If full,
try **(h(key) + 2) % TableSize**. If full,
try **(h(key) + 3) % TableSize**. If full…

Example: insert 38, 19, 8, 79, 10

Linear Probing

0	8
1	79
2	10
3	
4	
5	
6	
7	
8	38
9	19

Example: insert 38, 19, 8, 79, 10

38 % 10 =8

19 % 10 =9

8 % 10 =8 X
(8+1) % 10=9 X
(8+2) % 10=0

79 % 10 =9 X
(79+1) % 10=0 X
(79+2) % 10=1

10 % 10 =0 X
(10+1) % 10=1 X
(10+2) % 10=2

Quadratic Probing

How to deal with collisions?

If **h(key)** is already full,
try **(h(key) + 1^2) % TableSize**. If full,
try **(h(key) + 2^2) % TableSize**. If full,
try **(h(key) + 3^2) % TableSize**. If full…

Example :insert 89 , 18 , 49 , 58 , 79

Quadratic Example

0	49
1	
2	58
3	79
4	
5	
6	
7	
8	18
9	89

Example :insert 89 , 18 , 49 , 58 , 79

89 % 10 =9

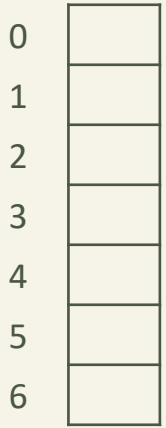
18 % 10 =8

49 % 10 =9 X
(49+1) %10 = 0

58 % 10 =8 X
(58+1) %10= 9 X
(58+4) % 10= 2

79 % 10 =9 X
(79+1) %10= 0 X
(79+4) % 10= 3

Try to



Example :insert 76 , 93 , 40 , 47 , 10 , 55

Table size : 7

Solve the collision with

- Linear probing
- Quadratic probing

0	47
1	55
2	93
3	10
4	
5	40
6	76

Example :insert 76 , 93 , 40 , 47 , 10 , 55

Table size : 7

Linear probing

76 % 7 =7
93 % 7 =2
40 % 7 = 5

47 % 7 = 5 X
(47+1) % 7 = 6 X
(47+2) % 7 = 0

10 % 7 = 3

55 % 7 = 6 X
(55+1) % 7 =0 X
(55+2) % 7 = 1

Quadratic probing

76 % 7 =7
93 % 7 =2
40 % 7 = 5

47 % 7 = 5 X
(47+1) % 7 = 6 X
(47+4) % 7 = 2 X
(47+9) % 7 = 0

10 % 7 = 3

55 % 7 = 6 X
(55+1) % 7 =0 X
(55+4) % 7 = 3 X
(55+9) % 7 = 1

Thanks ☻

Eng / Hajar Nagm