Hashing. (Hashing is a technique used for searching purpose)

- Linear Search → TC: O(N)
- Binary Search → TC: O($\log_2 N$)

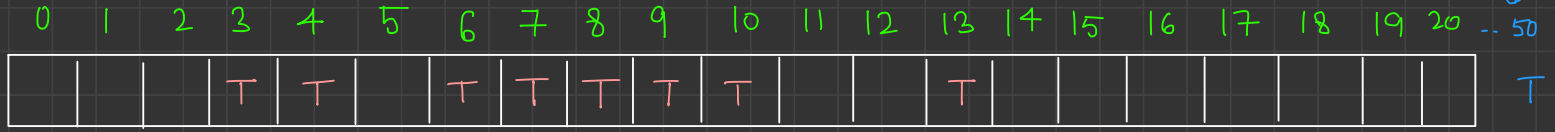
→ TC: O(1) Searching.

ele : 8, 3, 13, 6, 4, 10, 9, 7, 50 (user input)

key = 13

search(int val) ~> TC : O(1)

search(13)



arr

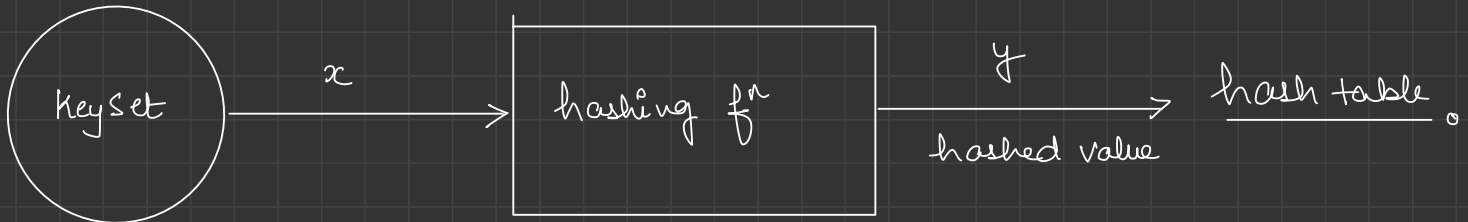
search(16)

search

```
if (arr[key] == true)
    return true;
else
    return false
```

A lot of memory was wasted!

* Hence, to over come this hashing was introduced.



hashing fn

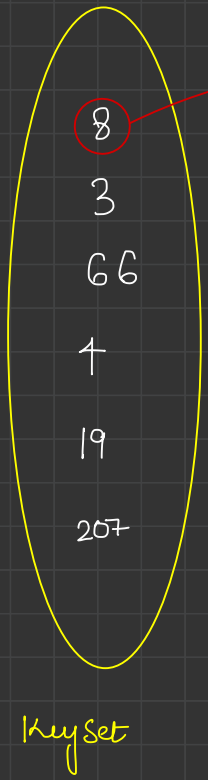
step1 $g(x) = k$

- $g(x)$ → key
- k → integer value

step2 $h(k) = y$

- y → hashed value

$h(x) = x \% 10$ hash fn



$h(8) = 8 \% 10 = 8$ → hashed value

$h(3) = 3 \% 10 = 3$

$h(66) = 66 \% 10 = 6$

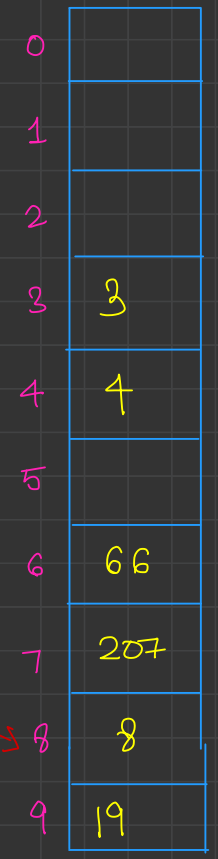
$h(4) = 4 \% 10 = 4$

$h(19) = 19 \% 10 = 9$

$h(207) = 207 \% 10 = 7$

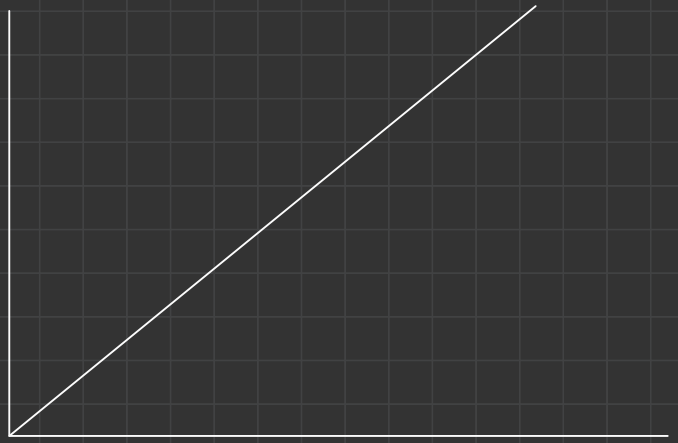
search (207)

→ TC: O(1)

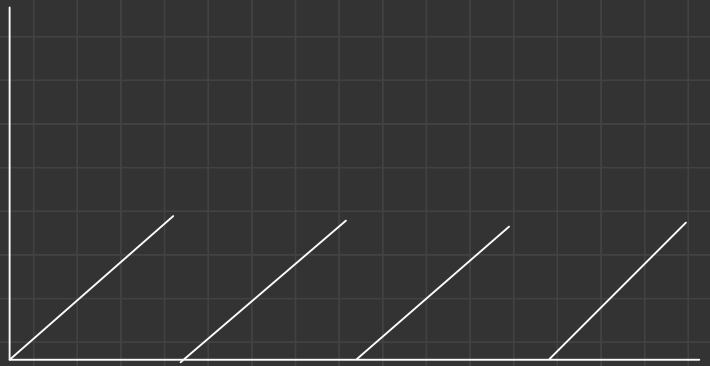


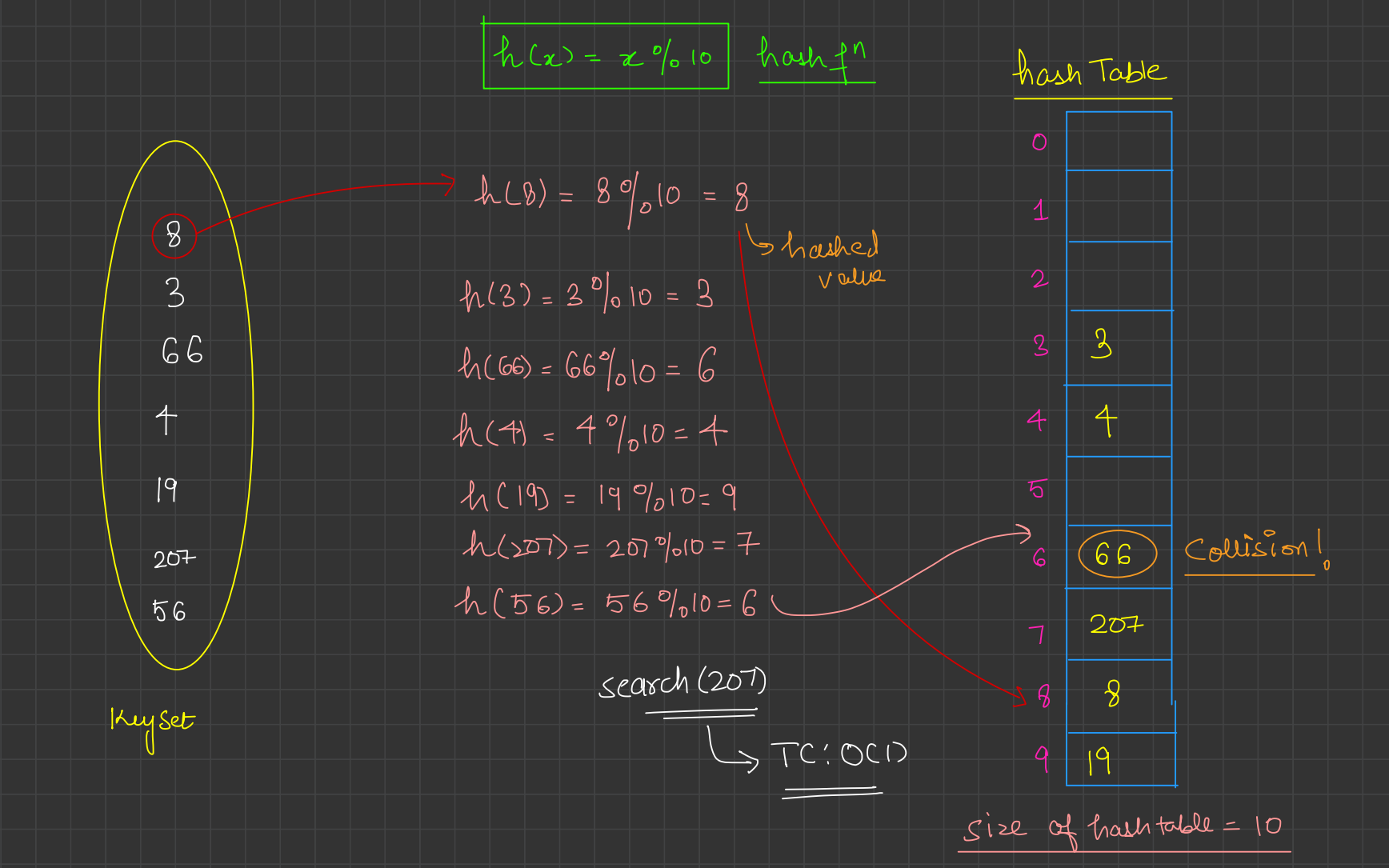
size of hash table = 10

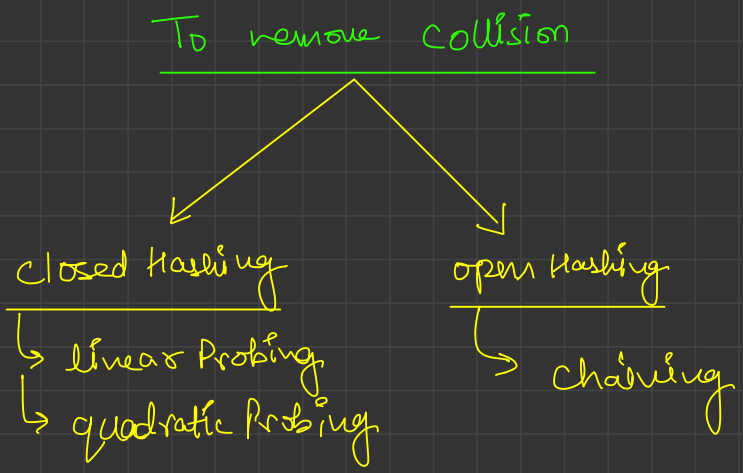
One to one mapping



many to one mapping




h(x) = x % 10
hash fn
hash Table
Key set
8
3
66
4
19
207
56
h(8) = 8 % 10 = 8
hashed value
h(3) = 3 % 10 = 3
h(66) = 66 % 10 = 6
h(4) = 4 % 10 = 4
h(19) = 19 % 10 = 9
h(207) = 207 % 10 = 7
h(56) = 56 % 10 = 6
Collision!
search(207)
TC: O(1)
size of hash table = 10


To remove collision
closed Hashing
Linear Probing
quadratic Probing
open Hashing
chaining

Chaining

$h(x) = x \% 10$ hash fn

Key set: 8, 3, 66, 4, 19, 207, 56, 6

$h(8) = 8 \% 10 = 8$ → hashed value

$h(3) = 3 \% 10 = 3$

$h(66) = 66 \% 10 = 6$

$h(4) = 4 \% 10 = 4$

$h(19) = 19 \% 10 = 9$

$h(207) = 207 \% 10 = 7$

$h(56) = 56 \% 10 = 6$

$h(6) = 6 \% 10 = 6$

search(207)

→ TC: O(1)

Hash Table

0	
1	
2	
3	3
4	4
5	
6	66 → 56 → 6
7	207
8	8
9	19

Size of hash table = 10

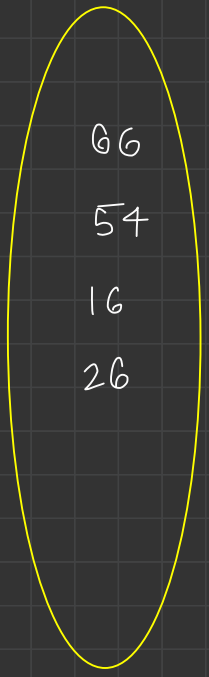
✓ size of hashtable = 75 % of Range of i/p

Linear Probing

$$h(x) = [h'(x) + g(i)] \% 10$$ hash fn

$h'(x) = x \% 10$

$g(i) = i$, $i \rightarrow 0, 1, 2 \ldots\ldots$



Key set

$h(66) = [h'(66) + g(i)] \% 10$

$h(66) = [6 + 0] \% 10 = 6$

$h(54) = [h'(54) + g(i)] \% 10$

$= [4 + 0] \% 10 = 4$

$h(16) = [h'(16) + g(i)] \% 10$

$= [6 + 0] \% 10 = 6$

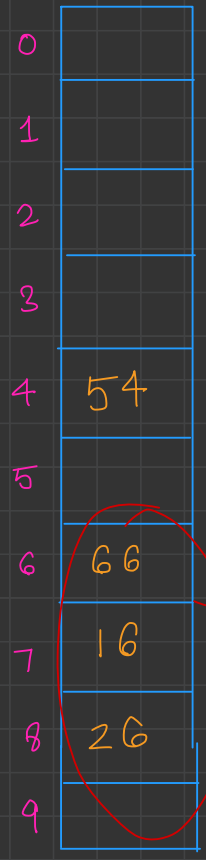
$= [6 + 1] \% 10 = 7$

$h(26) = [h'(26) + g(i)] \% 10$

$= [6 + 0] \% 10 = 6$ $= [6+2] \% 10$

$= [6 + 1] \% 10 = 7$ $= 8$

Hash Table



Size of hash table = 10

Quadratic Probing

$$h_i(x) = \left[h'(x) + g(i) \right] \% \text{ size}$$

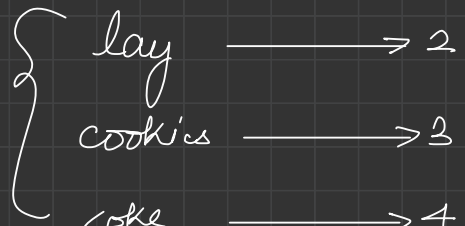
$$h'(x) = x \% \text{ size}$$

$$g(i) = i^2 \quad , \quad i \rightarrow 0, 1, 2, 3 \ldots\ldots$$

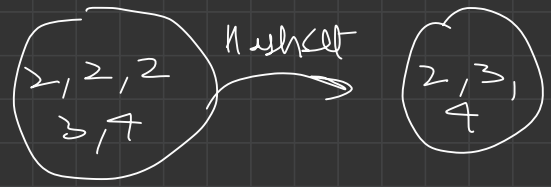
2 - Data structures

1. HashMap
2. HashSet

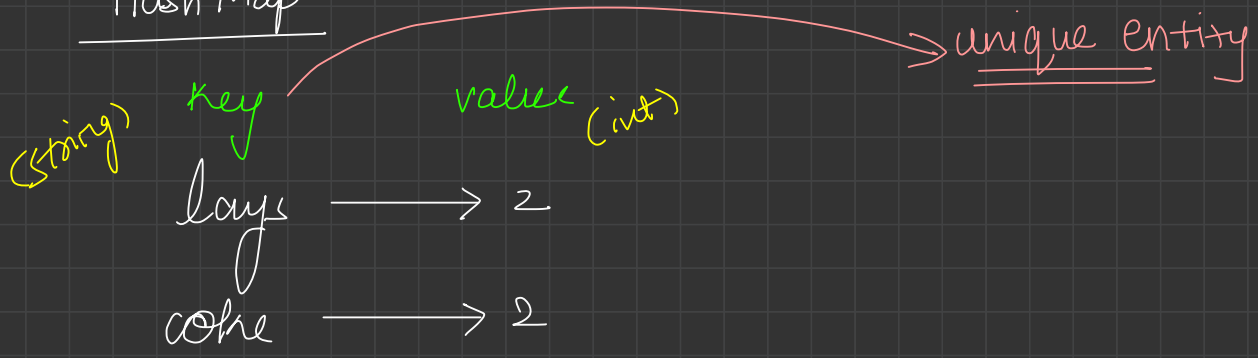
Key - value pairs



unique entity



Hash Map



FYI

TreeMap → Key is in a sorted format

TreeSet → value present are in sorted format

TC of each Method log(N)

↓

Searching, add, removing

red-Black tree