# Hashing

Searching → TC: O(1)

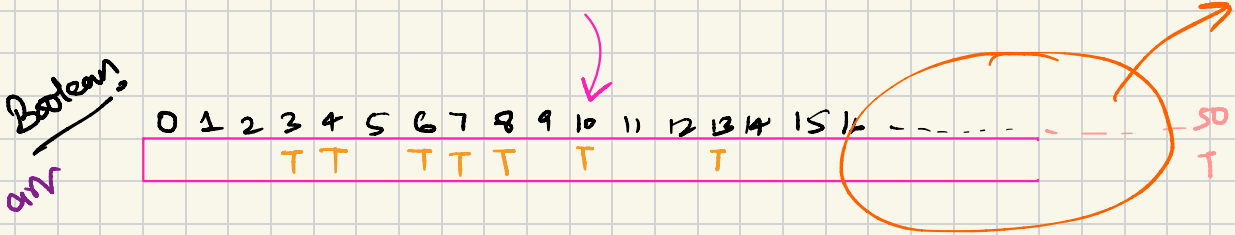
- a technique used for searching purposes

→ Linear Search
  → TC: O(N)

→ Binary Search
  → TC: O($\log_2 N$)

i/p user

8, 3, 13, 6, 7, 4, 10, 50




Memory Wastage

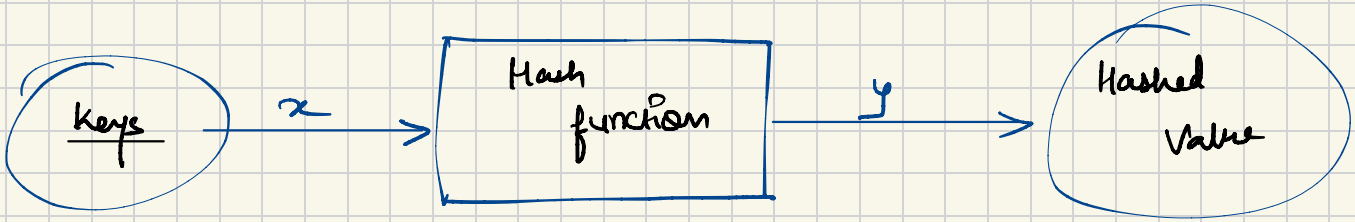
```
boolean search(int key)
{
    if (arr[key] == true)
        return true;
    else
        return false;
}
```

TC: O(1)
SC: O(1)

NOTE: NOT A GOOD APPROACH AS HIGH MEMORY USAGE.

- HENCE HASHING WAS INTRODUCED

## Basics mechanism of Hashing.




Hash function

Step 1 : $g(x) = k$ (key → integer value)

Step 2 : $h(k) = y$ (y → hashed value)

Key Space

8
3
13
6
4
10

Hash fn

$h(k) = k$

$h(8) = 8$

$h(3) = 3$

$h(13) = 13$

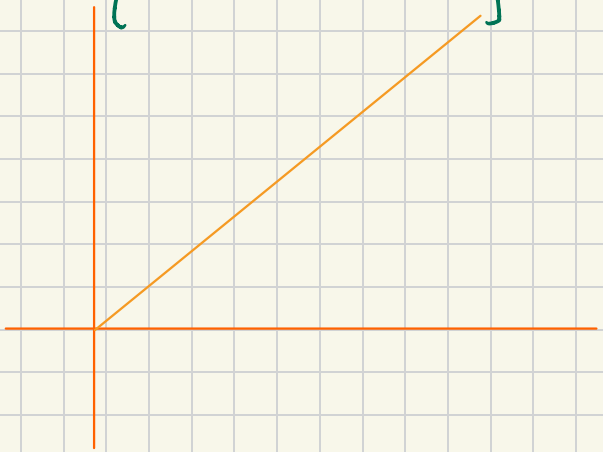
$h(6) = 6$

$h(4) = 4$

$h(10) = 10$

One to one fn

{MEMORY WASTAGE}



hash Table

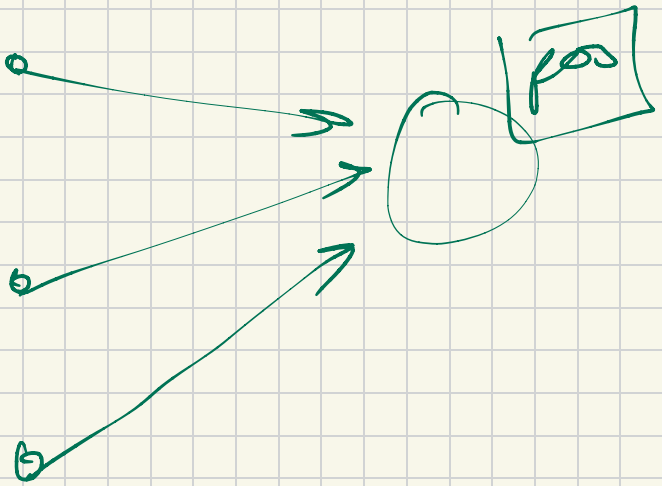
| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | 3 |
| 4 | 4 |
| 5 | |
| 6 | 6 |
| 7 | |
| 8 | 8 |
| 9 | |
| 10 | 10 |
| 11 | |
| 12 | |
| 13 | 13 |
| 14 | |
| 15 | |
| 16 | |

one - one X

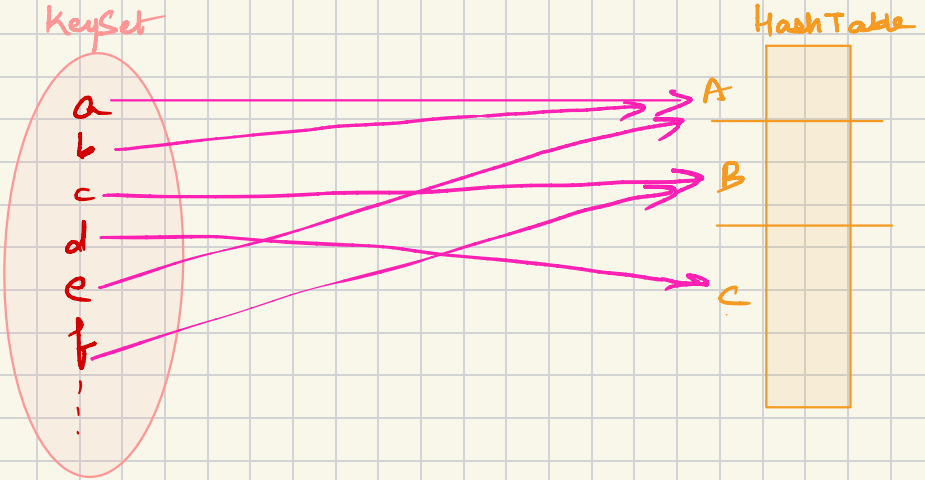
one - many X

many - one ✓

Reduce



## Many to One

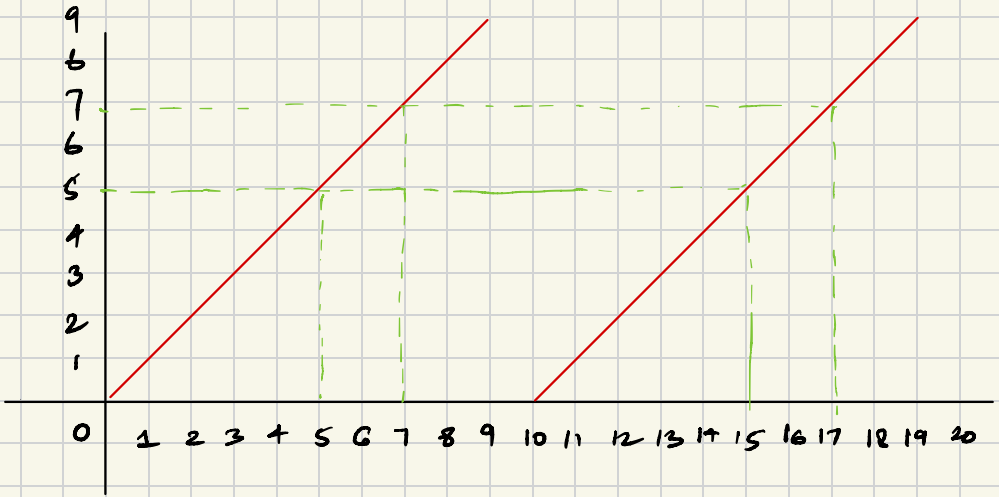


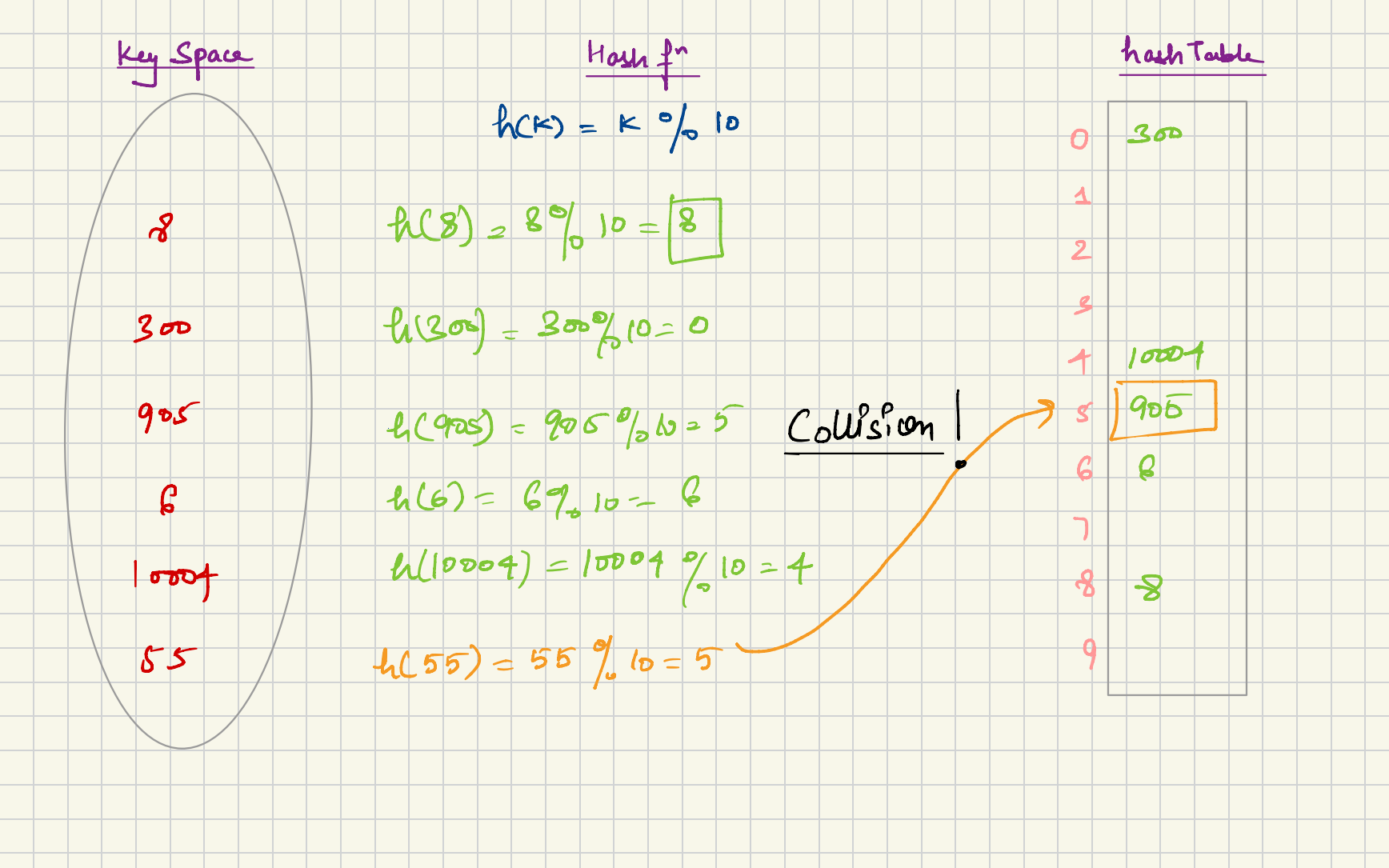
hash fun^n

[0, 9] → Remainder

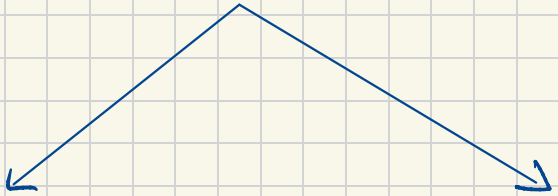
$h(K) = K \% 10$

Many – one f^n




Key Space
8
300
905
6
10004
55
Hash fn
h(K) = K % 10
h(8) = 8 % 10 = 8
h(300) = 300 % 10 = 0
h(905) = 905 % 10 = 5
h(6) = 6 % 10 = 6
h(10004) = 10004 % 10 = 4
h(55) = 55 % 10 = 5
Collision !
hash Table
0 300
1
2
3
4 10004
5 905
6 6
7
8 8
9

# Methods to Remove Collision

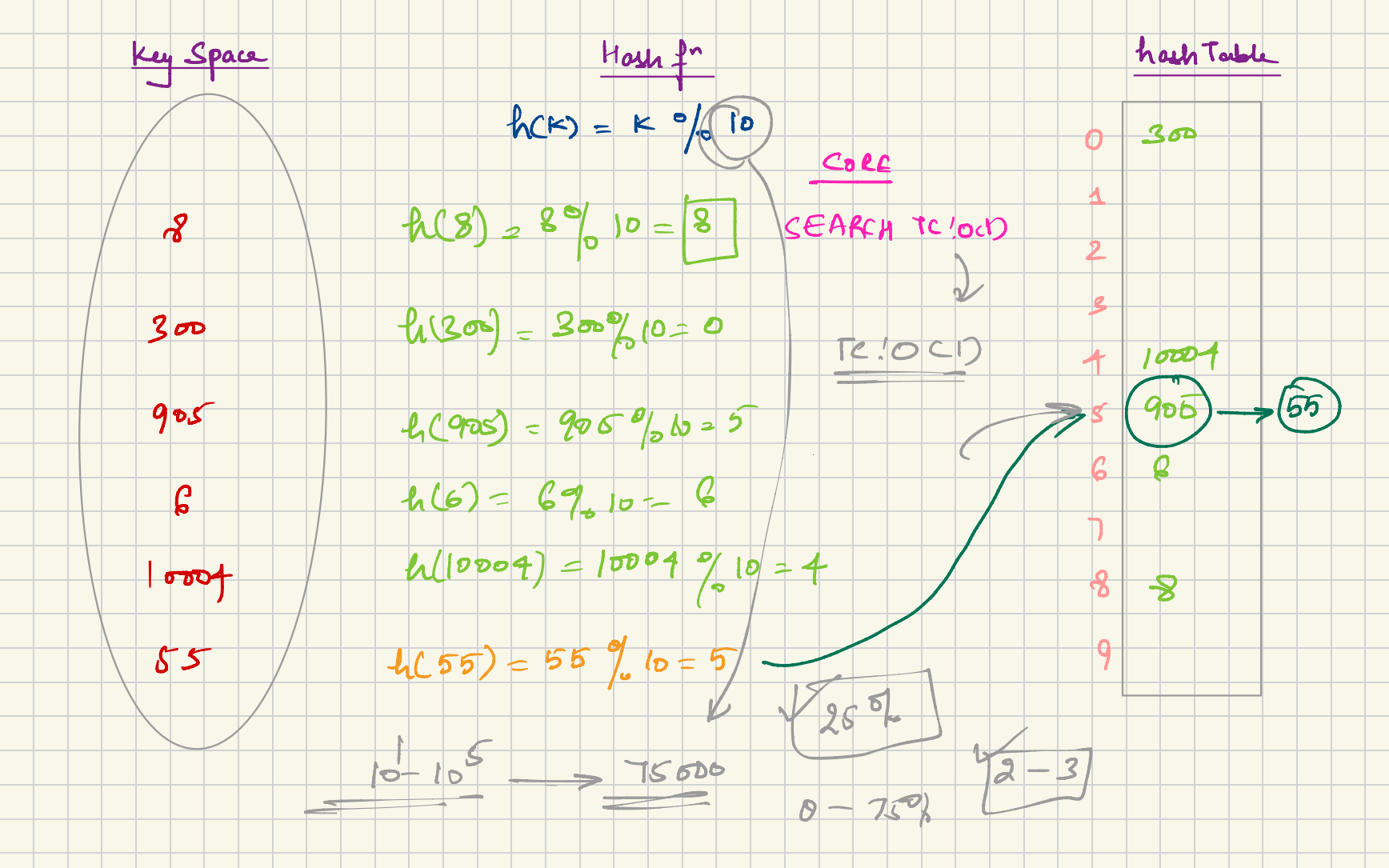


## Open Hashing

- Chaining

## closed Hashing

- Linear probing
- Quadratic probing

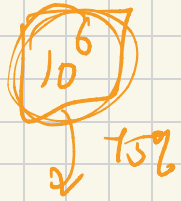

Key Space
8
300
905
6
10004
55
Hash fn
h(K) = K % 10
h(8) = 8 % 10 = 8
h(300) = 300 % 10 = 0
h(905) = 905 % 10 = 5
h(6) = 6 % 10 = 6
h(10004) = 10004 % 10 = 4
h(55) = 55 % 10 = 5
10^1 - 10^5
75000
CORE
SEARCH TC:O(1)
TC:O(1)
25%
0 - 75%
2 - 3
hash Table
0
1
2
3
4
5
6
7
8
9
300
10004
905
55
6
8

## Load factor

$h(k) = \text{key } \% \text{ (Size of HashTable)}$

Size of HashTable = 75% of Range

Size of HashTable ✓ L.F.




7500000

Key Space

8

300

905

6

10004

55

Hash fn

$h'(k) = \{h(k) + f(i)\} \% \text{size}$

$h(k) = k \% \text{size}$

$f(i) = i \quad [0, 1, 2, 3, \ldots]$

$h'(8) = \{h(8) + f(0)\} \% 10$

$= \{8 + 0\} \% 10$

$= 8$

$h'(300) = 0$

$h'(905) = 5$

$h'(6) = 6$

$h'(10004) = 4$

$h'(55) = (5+0) \% 10 = 5$

$= (5+1) \% 10 = 6$

$= (5+2) \% 10 = 7$

hash Table

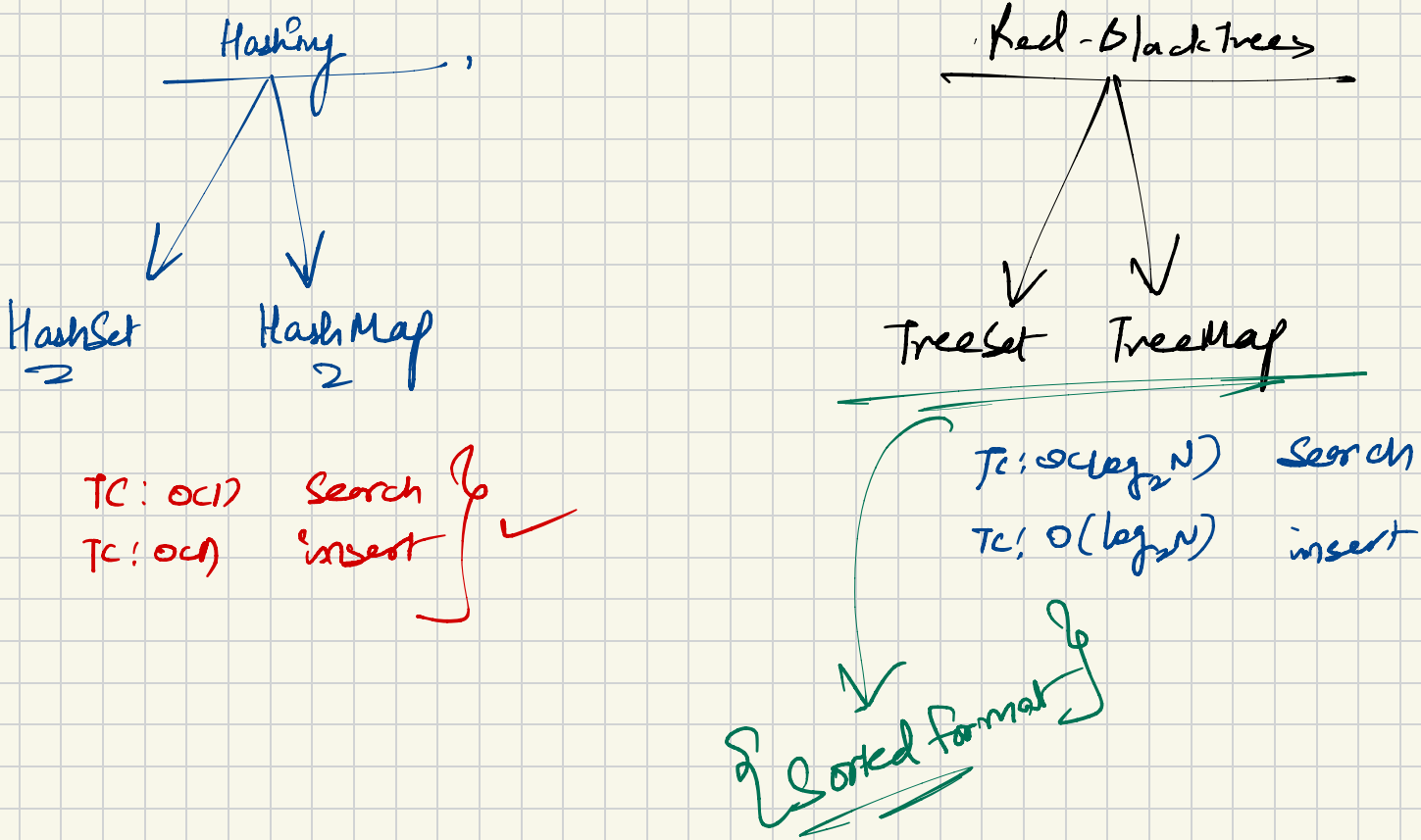
| | |
|---|---|
| 0 | 300 |
| 1 | |
| 2 | |
| 3 | |
| 4 | 10004 |
| 5 | 905 |
| 6 | 6 |
| 7 | 55 |
| 8 | 8 |
| 9 | |

# quadratic Probing

$$h'(k) = \{h(k) + f(i)\} \% \text{ size}$$

$$h(k) = k \% \text{ size}$$

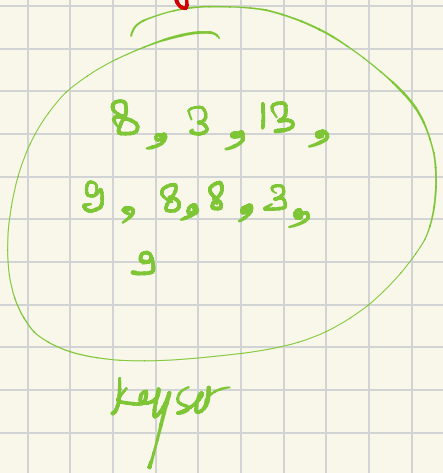
$$f(i) = i^2 \; ; \; i \in 0, 1, 2 \ldots$$

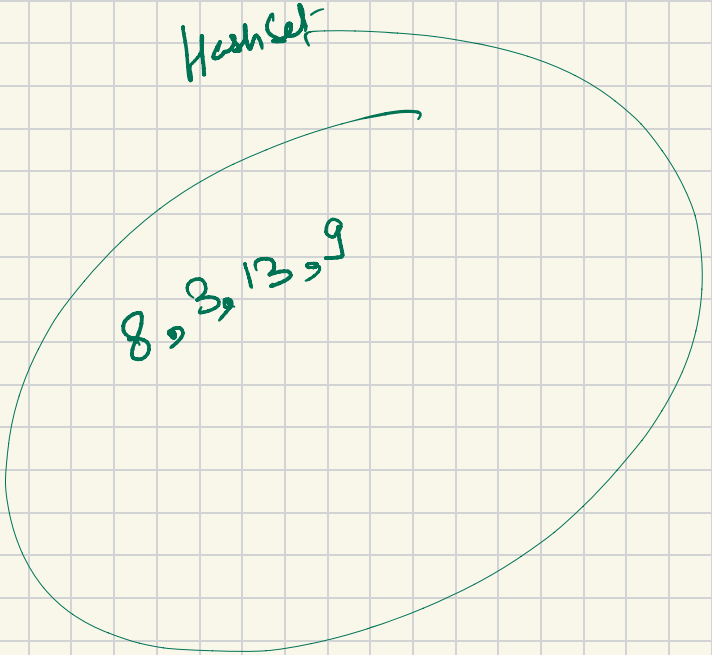

Hashing
HashSet
HashMap
TC : O(1) Search
TC : O(1) insert
Red-Black trees
TreeSet
TreeMap
TC : O(log2N) Search
TC : O(log2N) insert
Sorted format

HashSet

→ stores unique Entities

→ searching O(1)



# HashMap

DS to store (key, value) pairs