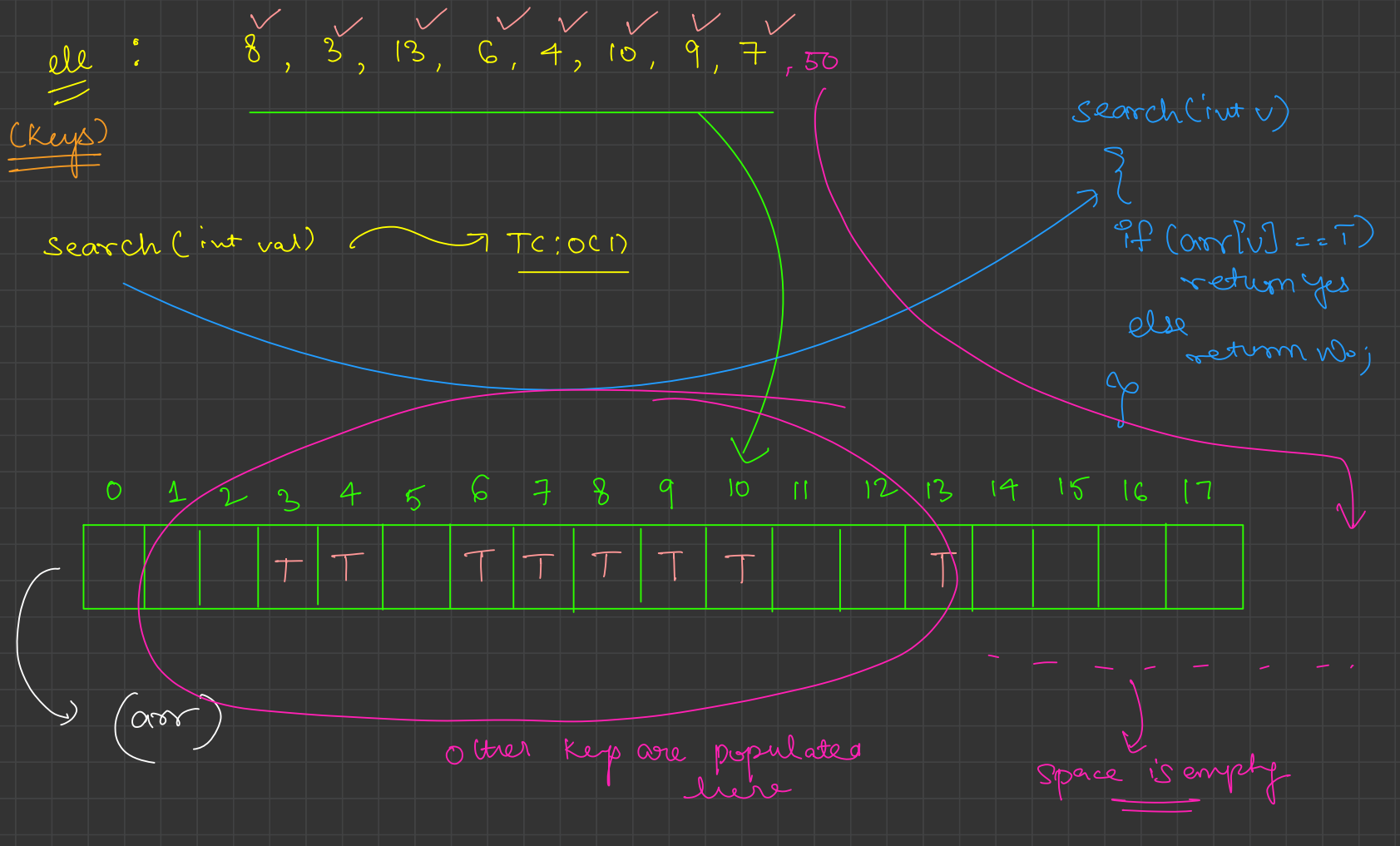
Hashing. (Hashing is a technique used for searching purpose)

- 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
(Keys)
search(int v)
{
if (arr[v] == T)
return yes
else
return No;
}
search(int val) → TC: O(1)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
T T T T T T T T
(arr)
Other Keys are populated here
Space is empty

In above eg: a lot memory is required.

To improve this idea, hashing was introduced.

Hash Table

list of Elements we have is known as Key Space

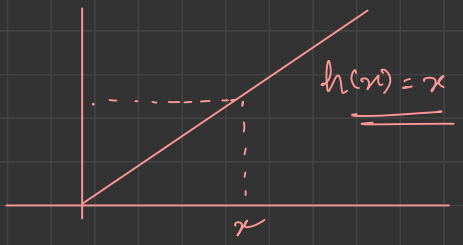
$h(8) =$

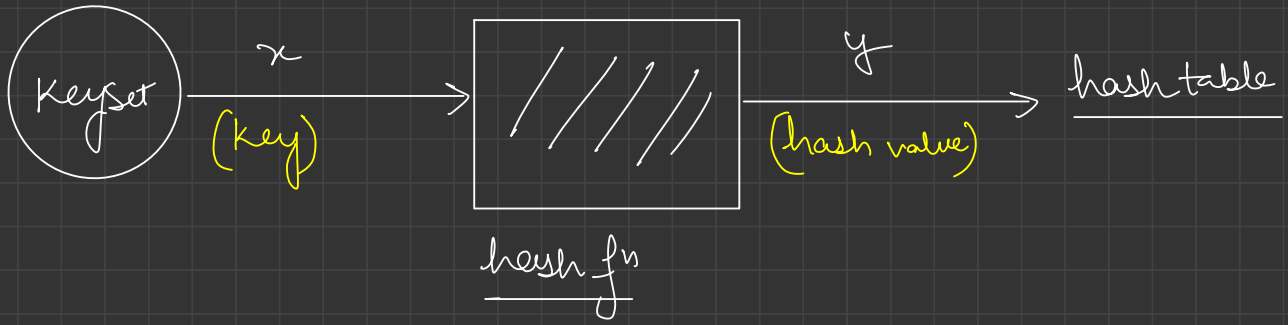
$h(x) = x$

8
3
13
6
4
10

hash fn

{one to one mapping}





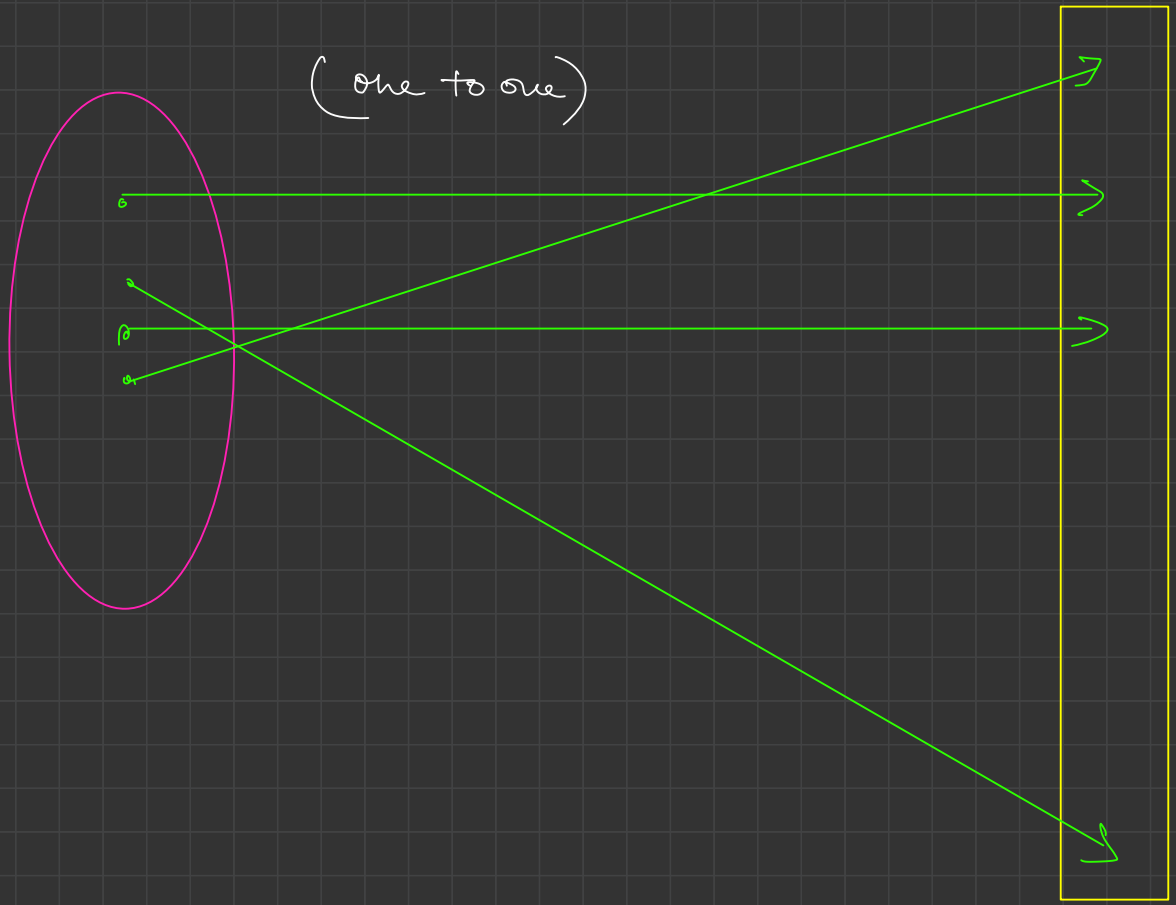
Step1 $g(x) = K$

key → integer value

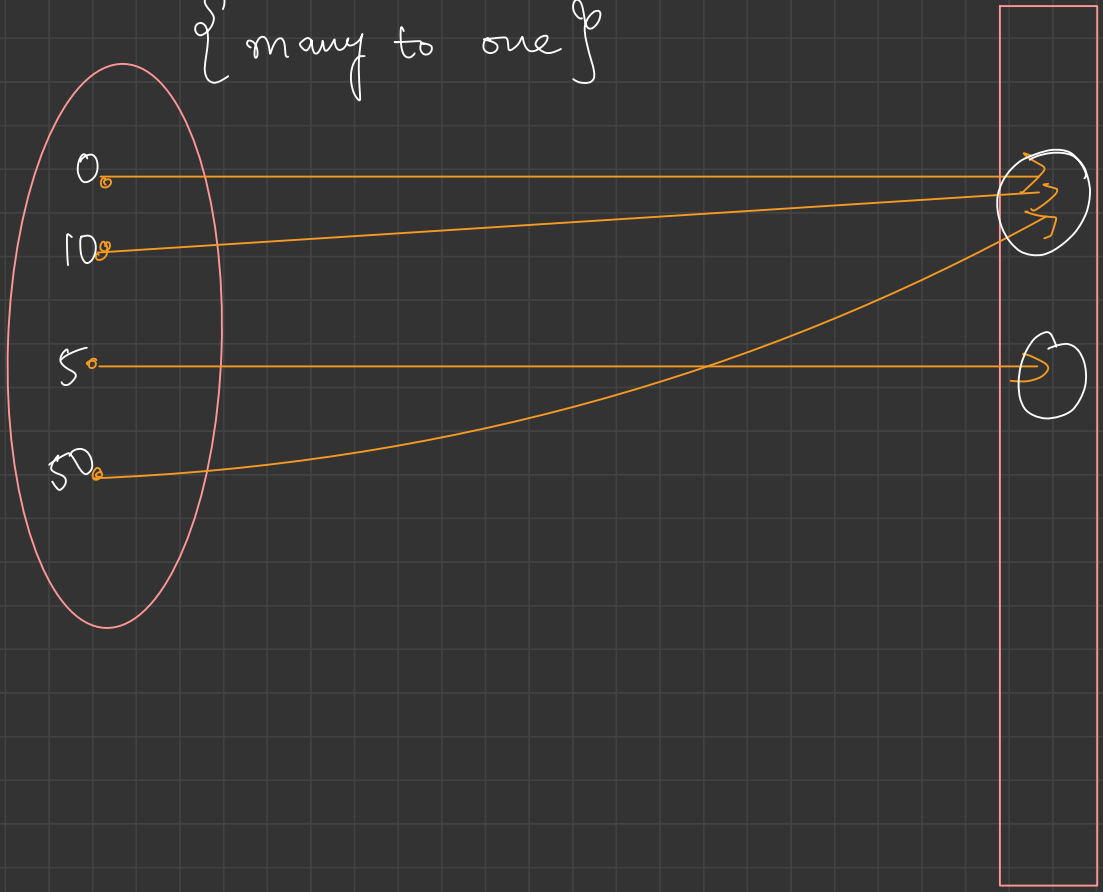
hashing fn

Step2 $h(K) = y$

(one to one)



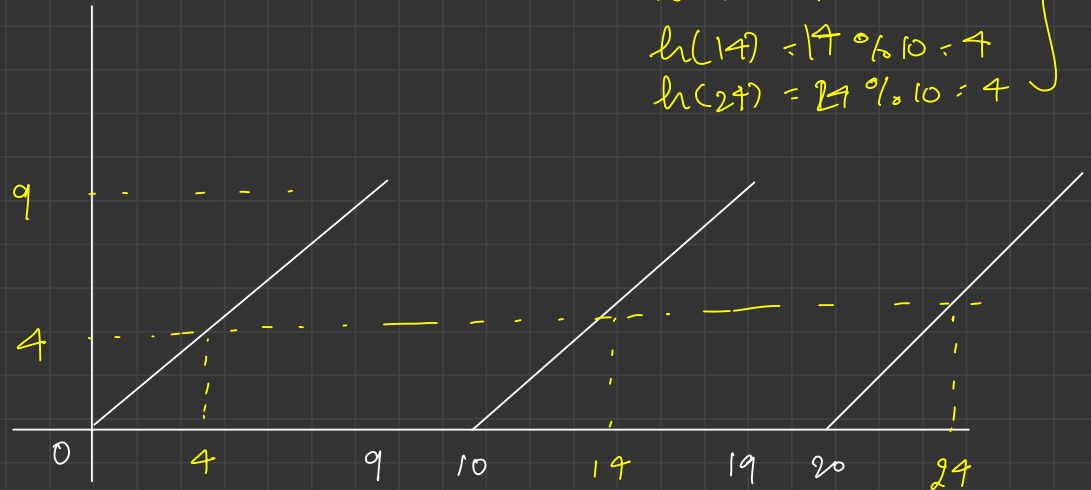
{many to one}

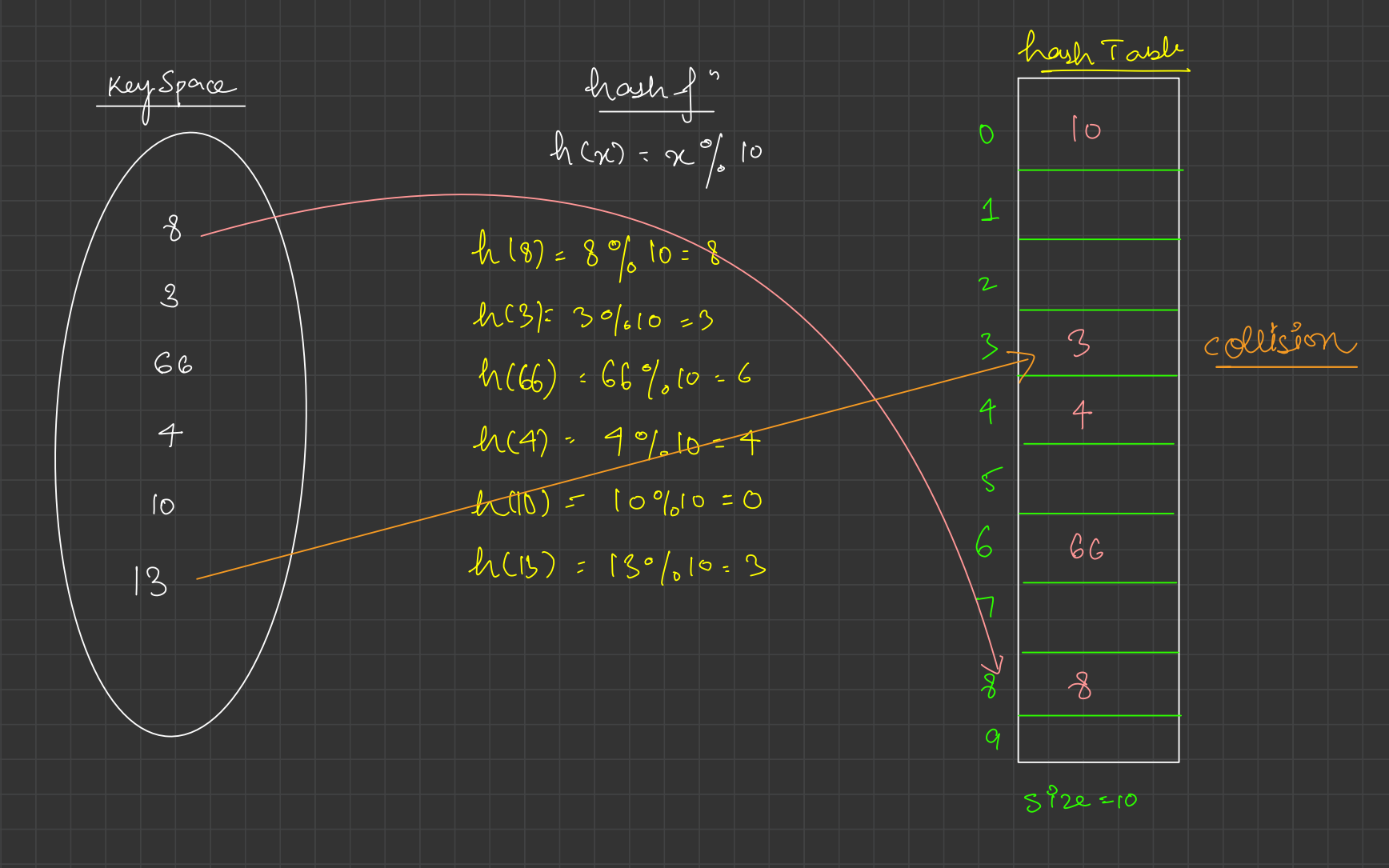


Many to one fn

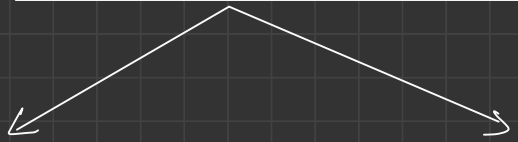
$$h(x) = x \% 10$$

$h(4) = 4 \% 10 = 4$
$h(14) = 14 \% 10 = 4$
$h(24) = 24 \% 10 = 4$




Key Space
8
3
66
4
10
13
hash f"
h(x) = x % 10
h(8) = 8 % 10 = 8
h(3) = 3 % 10 = 3
h(66) = 66 % 10 = 6
h(4) = 4 % 10 = 4
h(10) = 10 % 10 = 0
h(13) = 13 % 10 = 3
hash Table
0
1
2
3
4
5
6
7
8
9
10
3
4
66
8
collision
Size = 10

Methods to Remove Collision

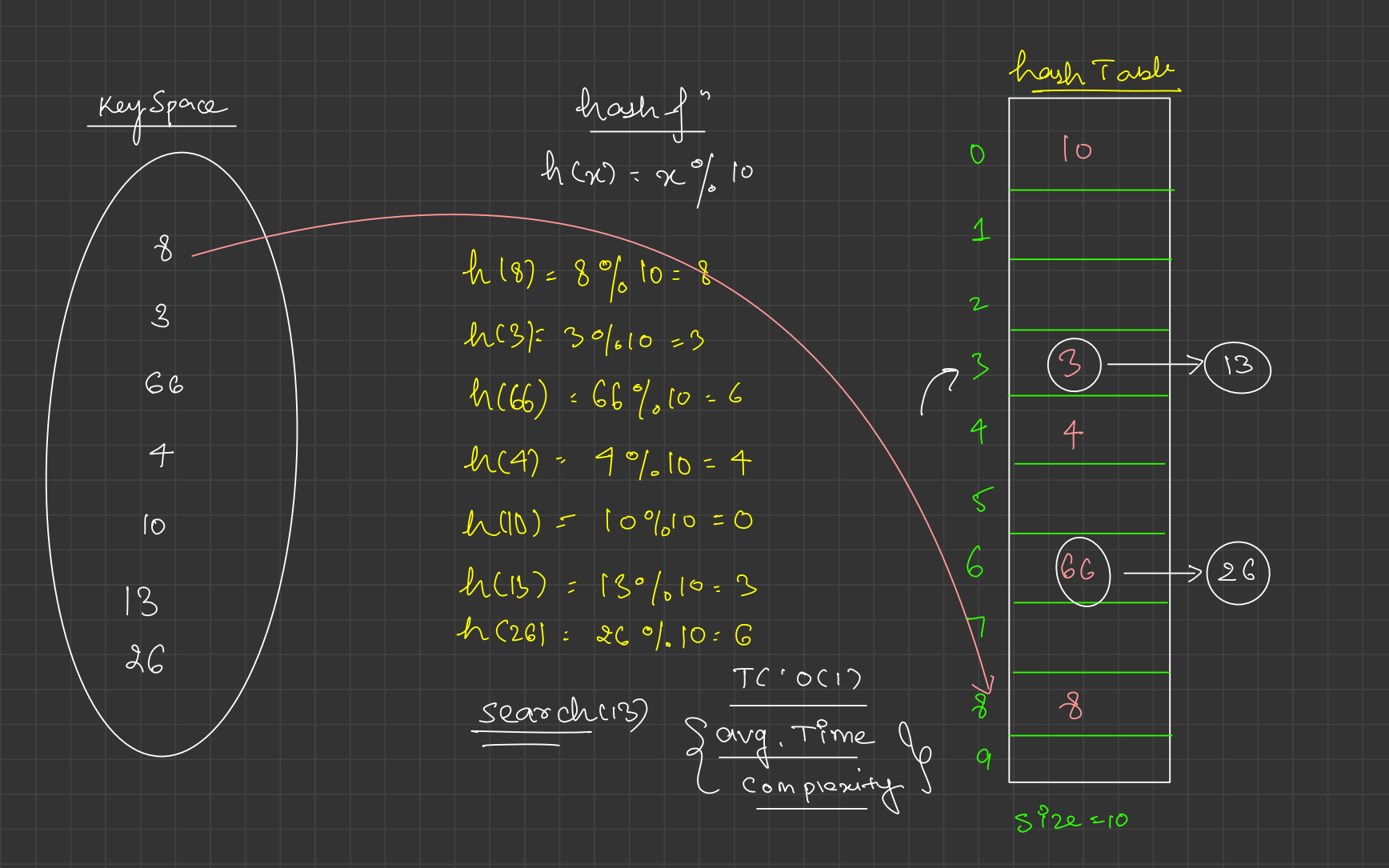


open hashing

1. chaining

closed hashing

1. linear probing
2. quadratic Probing


Key Space
8
3
66
4
10
13
26
hash f^n
h(x) = x % 10
h(8) = 8 % 10 = 8
h(3) = 3 % 10 = 3
h(66) = 66 % 10 = 6
h(4) = 4 % 10 = 4
h(10) = 10 % 10 = 0
h(13) = 13 % 10 = 3
h(26) = 26 % 10 = 6
search(13)
TC : O(1)
avg. Time Complexity
hash Table
0
1
2
3
4
5
6
7
8
9
10
3
13
4
66
26
8
size = 10

$h(x) = x \%$ size

→ 75% of keyspace range

{0 — 10,000}

$h(x) \cdot x\% \cdot 7500$

Key Space

8
3
66
4
10
13

hash f^n

$h(x) = x \% 10$

Linear Probing

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

$g(x) = i$, $i \to 0 \ldots\ldots \infty$

$h'(8) = [h(8) + g(8)] \% 10$
$= [8 + 0] \% 10 = 8$

$h'(3) = [h(3) + g(3)] \% 10$
$= [3 + 0] \% 10$

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

hash Table

0	10
1	
2	
3	3
4	4
5	13
6	66
7	
8	8
9	

Size = 10

$h'(4) = [h(4) + g(4)] \% 10$

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

$h'(10) = [h(10) + g(1)] \% 10$

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

$h'(13) = [h(13) + g(13)] \% 10$

$= [3 + 0] \% 10 = 3$

$= [h(13) + g(13)] \% 10$

$= [3 + 1] \% 10$

$= 4$

$= [h(13) + g(13)] \% 10$

$= [3 + 2] \% 10$

$= 5$

quadratic hashing

$$h'(x) = [h_1(x) + g(x)] \% \text{ size}$$

$$g(i) = i^2, \quad 0, 1, 4, 9, 16, 25, \ldots$$

2 - Data Structures

① HashMap
② HashSet
→ Hash Theory

TC: O(1)

Key value pairs

Rohan → 80
Rahul → 20
Jacob → 80

① TreeMap → Keys asc order
② TreeSet → Value in asc. order
→ Red-Black Tree

TC: O(log N)

Stores unique values