Hashing → technique used for searching purpose

Linear Search → TC: $O(N)$

Binary Search → TC: $O(\log_2 N)$

$\boxed{TC: O(1)}$ → searching

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

Key = 13

search (int val)

→ TC: O(1)

Boolean arr[]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	---	50
			✓	✓		✓	✓	✓	✓	✓			✓							✓

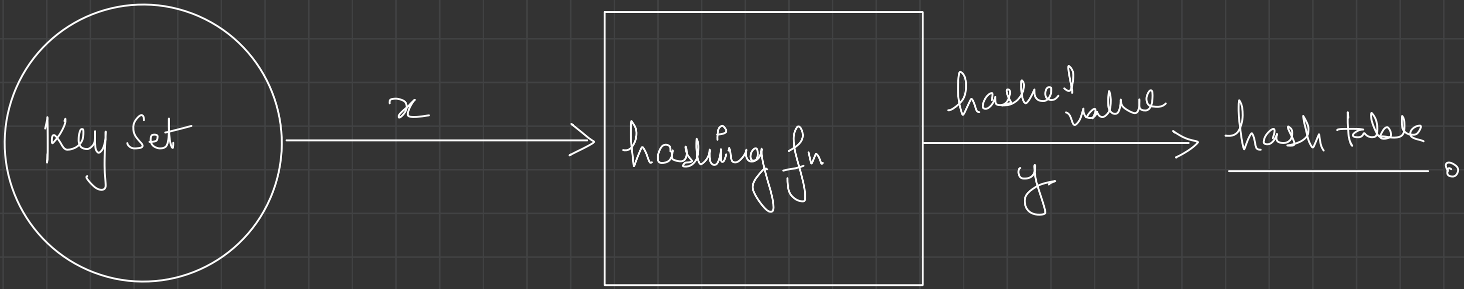
TC: O(1)

```
search(13)
{
    if (arr[13] == true)
        return true;
    else
        return false;
}
```

a lot of emptyspace

Memory wastage!

to overcome issue, hashing was introduced.



hashing f^n (2 step process)

Step 1 $g(x) = K$ integer value

Step 2 $h(K) = y$ hashed value

$\boxed{h(x) = x \% 10}$ hash f^n

10 → size of hash table

Keyset

- 8
- 3
- 66
- 4
- 19
- 207

$h(8) = 8 \% 10 = 8$

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

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

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

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

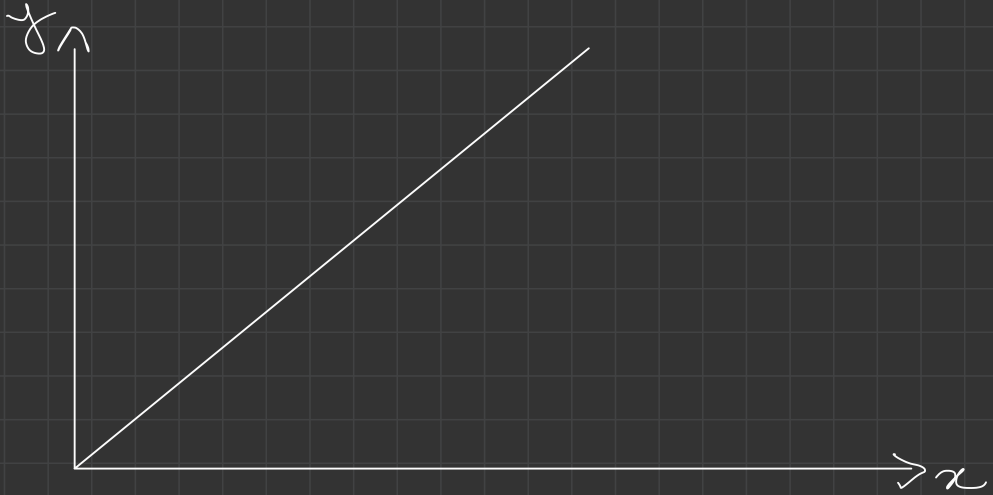
$h(207) = 207 \% 10 = 7$

Hash Table

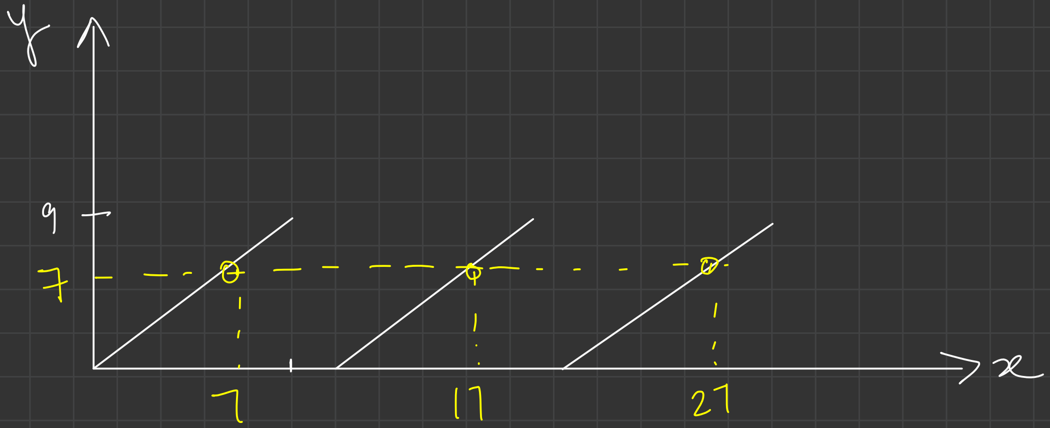
0	
1	
2	
3	3
4	4
5	
6	66
7	207
8	8
9	19

search (207) → TC: O(1)

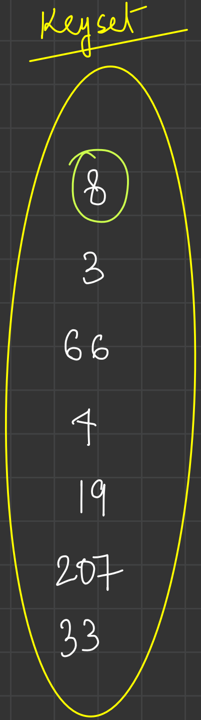
One to One Mapping



many to one



$\boxed{h(x) = x \% 10}$ hash f^n



$h(8) = 8 \% 10 = 8$

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

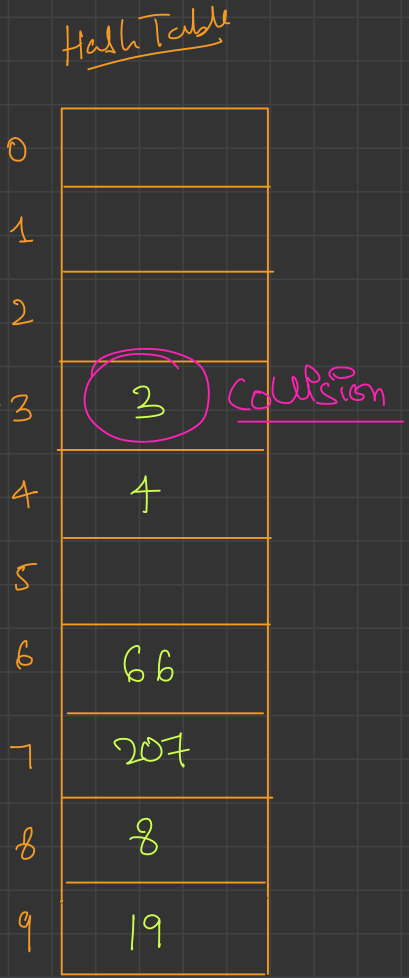
$h(66) = 66 \% 10 = 6$

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

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

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

$h(33) = 33 \% 10 = 3$



Methods to Remove Collision

Closed hashing

- → linear Probing
- → quadratic Probing

Open hashing

- ↳ chaining

Chaining.

$h(x) = x \% 10$ hash fⁿ

Keyset

8

3

66

4

19

207

33

23

$h(8) = 8\% 10 = 8$

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

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

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

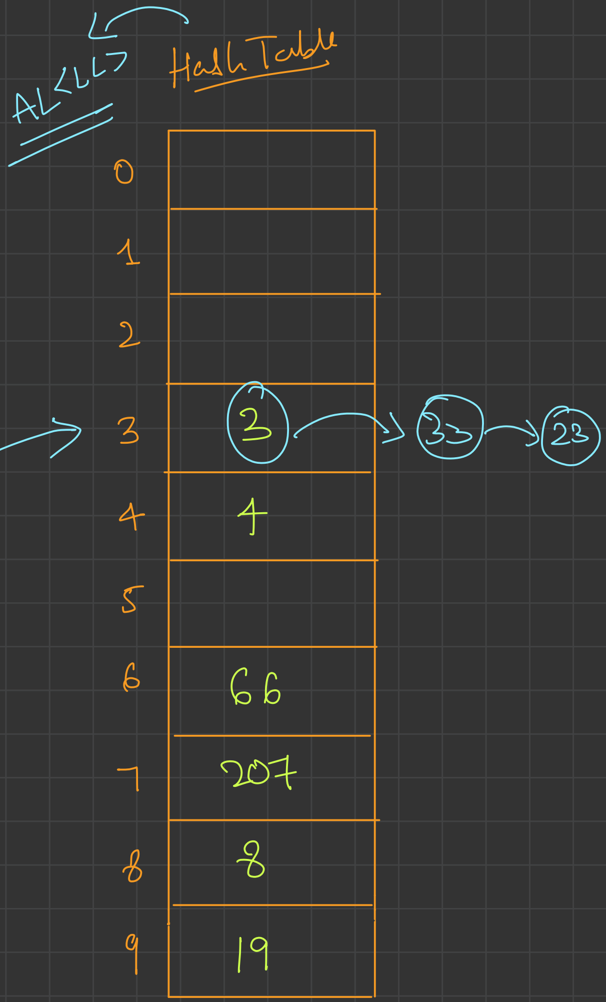
$h(19) = 19\% 10 = 9$

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

$h(33) = 33\% 10 = 3$

$h(23) = 3$

Searching (33) → 3



hash fⁿ

$h(x) = x \% \underline{\underline{\text{load factor}}}$

↓

75% of range of values given

$\underline{\underline{0 - 150}}$

↳ $lf = \boxed{75}$ ✓

Linear Probing

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

↳ (10)

Keyset

- 8
- 3
- 66
- 33
- 4
- 23

$h(x) = x \% 10$

$f(i) = i \,,\quad i \in R[0, \infty)$

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

$= (8+0) \% 10 = 8$

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

$= (3+0) \% 10 = 3$

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

$= (6+0) \% 10 = 6$

$h'(33) = [h(33) + f(0)] \% 10 = (3+0) \% 10 = 3$

$h'(33) = [h(33) + f(1)] \% 10 = (3+1) \% 10 = 4$

$h'(4) = [h(4) + f(0)] \% 10 = (4+0) \% 10 = 4$

fill → 5

Hash Table

0	
1	
2	
3	3
4	33
5	4
6	66
7	23
8	8
9	

linear hashing

↳ dis adv : clustering issue !

quad probing

$$\boxed{\begin{array}{l} h'(x) = [h(x) + f(i)] \,\%\, \text{size} \\[1em] h(x) = x \,\%\, \text{size} \\[1em] f(i) = i^2 \quad , \ i \in R \end{array}}$$

2 - Data Structures

1. HashSet
2. Hash Map

Based On hash Theory

TC: O(1)

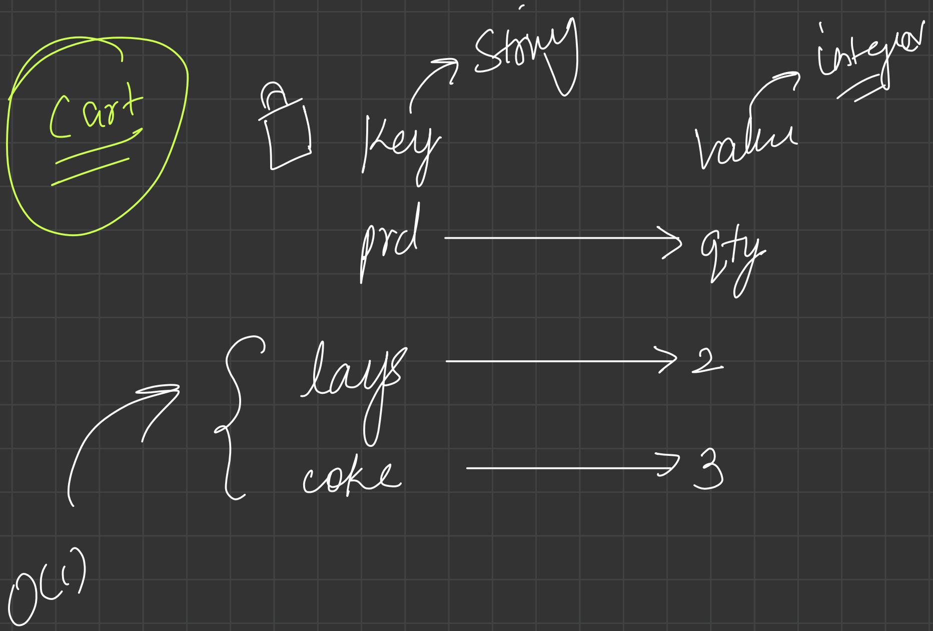
↓

Searching

HashSet : Stores unique entity

HashMap

↳ <key, value>



HashMap

TreeMap

TreeSet

here key are in ordered formate

Searching

TC: O(logN)

based

Red Black Tree

Self Balancing Tree