Hashing.

# it is a technique for searching purpose

① Linear Search

3	10	1	5	8	7	12	-10

TC: O(N) ~> searching

② Binary Search → {array should sorted}

{TC: O(log n) ~> searching}

③ Hashing → { TC: O(1) Searching }

{ 8, 3, 13, 6, 4, 10 }, 50

target = 4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…	50
A				3	4		6		8		10			13			50

memory wasted

search (4) → key

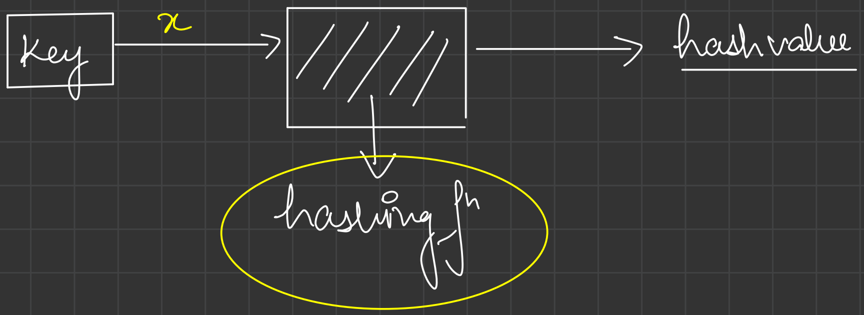
TC: O(1)

```
if ( A[key] != null)
{
    return true;
}
else
    return false;
```

dis adv

↳ high memory is used.

↳ hashig is Produced.



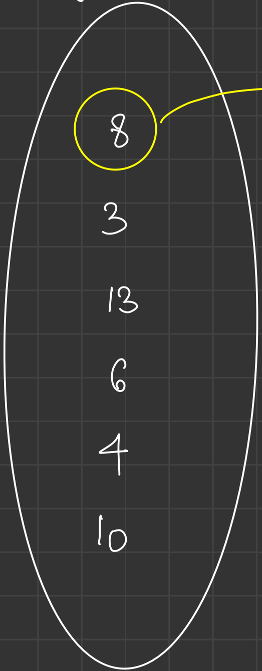
step1

$$\underset{\text{key}}{\underset{\downarrow}{g(x)}} = \underset{\text{integer value}}{\underset{\downarrow}{K}}$$

step2

$$\underset{\text{hashing } f^n}{\underset{\downarrow}{h}}(K) = y \rightarrow \boxed{\text{hash value}}$$

Key Space



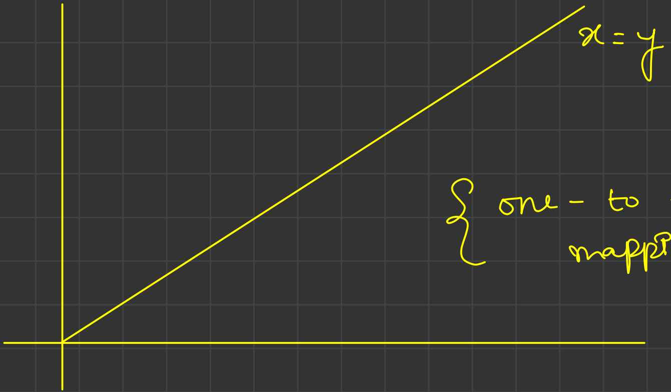
hashing f^n $\boxed{h(K) = y}$ hash value

$h(x) = x$

$h(8) = 8$ → hash value

hash table

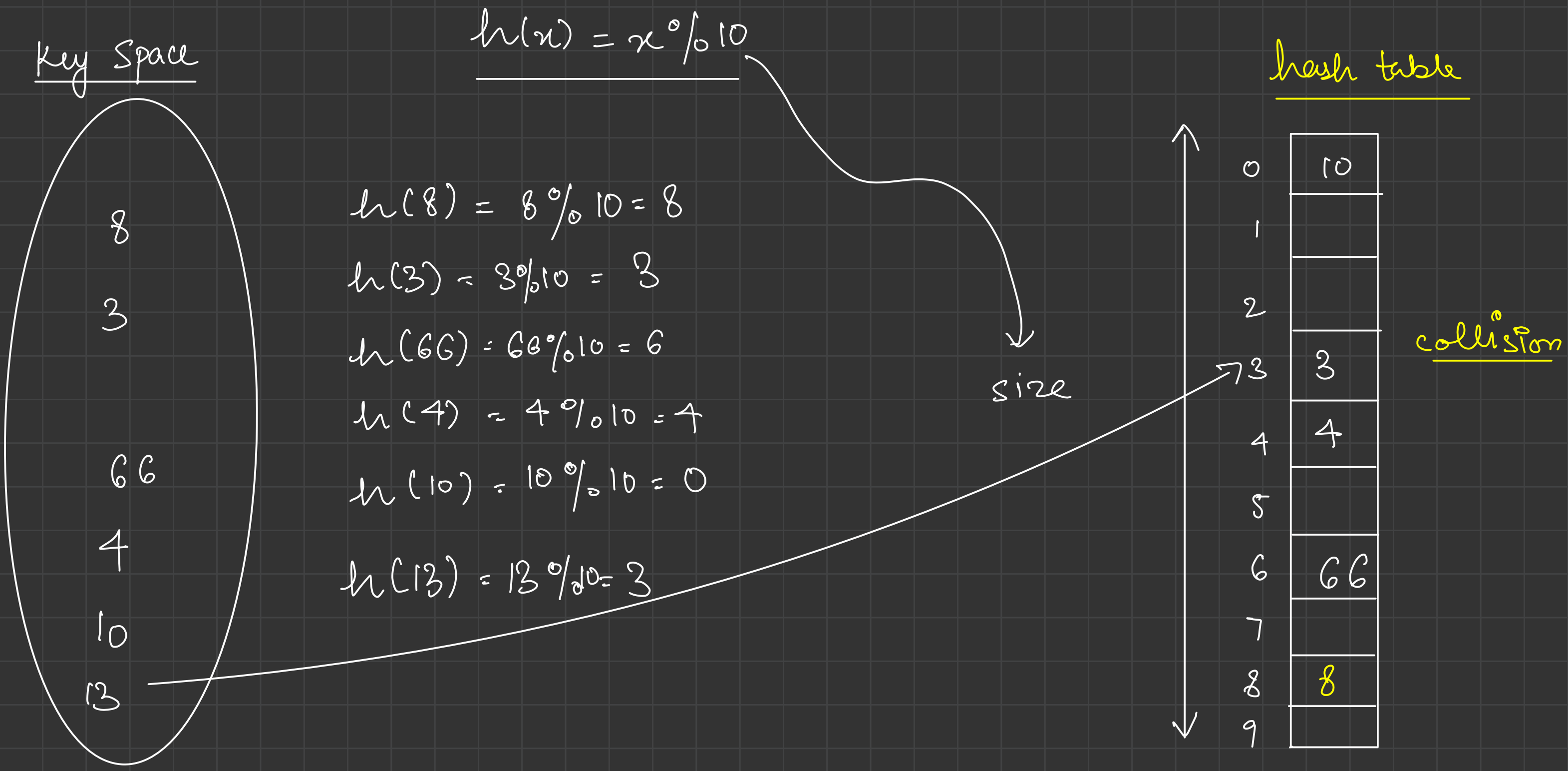
0	
1	
2	
3	
4	
5	
6	
7	
8	8
9	



{ one - to - one mapping

Many - to - One Mapping

$$\boxed{h(x) = x \% 10}$$



Key Space

8

3

66

4

10

13

$h(x) = x \% 10$

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

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

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

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

$h(10) = 10 \% 10 = 0$

$h(13) = 13 \% 10 = 3$

size

hash table

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

collision

Methods to Remove Collision

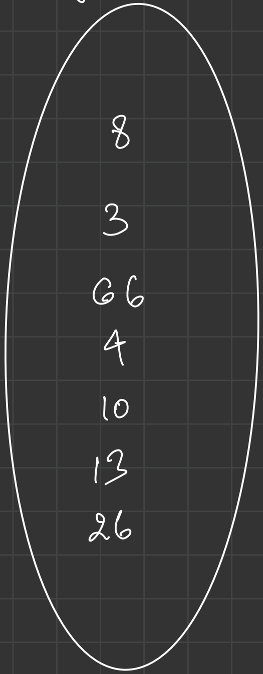
open hashing

- chaining

closed hashing

- linear Probing
- quadratic Probing

Key Space



$h(x) = x\%10$

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

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

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

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

$h(10) = 10\%10 = 0$

$h(13) = 13\%10 = 3$

$h(26) = 26\%10 = 6$

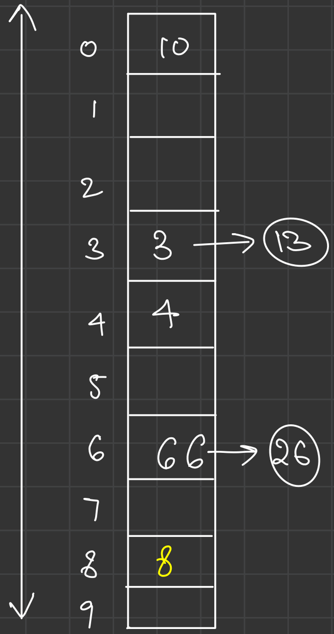
search(13)

TC : O(1)

searching

avg. time complexity

hash table



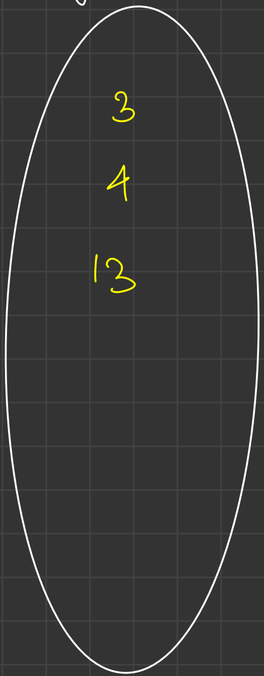
$0 \longrightarrow \boxed{10,000} \longrightarrow$ 75% of Range

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

linear probing

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

Key Space



$$h(x) = \{ h'(x) + f(i) \} \% 10$$

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

$f(i) = i$, 0, 1, 2, 3 . . .

$$h(3) = \{ h'(3) + f(i) \} \% 10$$

$$= \{3 + 0\} \% 10 = 3$$

$$h(4) = \{ 4 + 0 \} \% 10 = 4$$

$$h(13) = \{ 3 + 0 \} \% 10 = 3$$

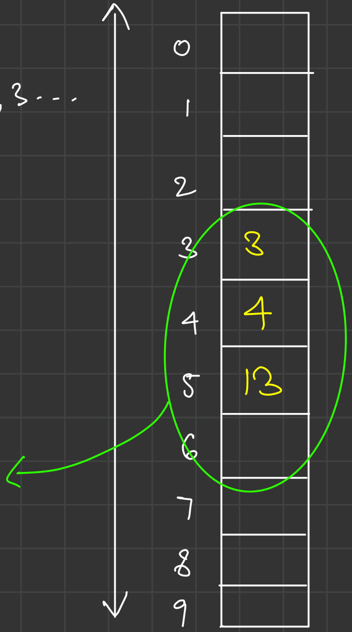
$$\{3+1\} \% 10 = 4$$

$$\{3+2\} \% 10 = 5$$

dis adv

clustering issues

hash table



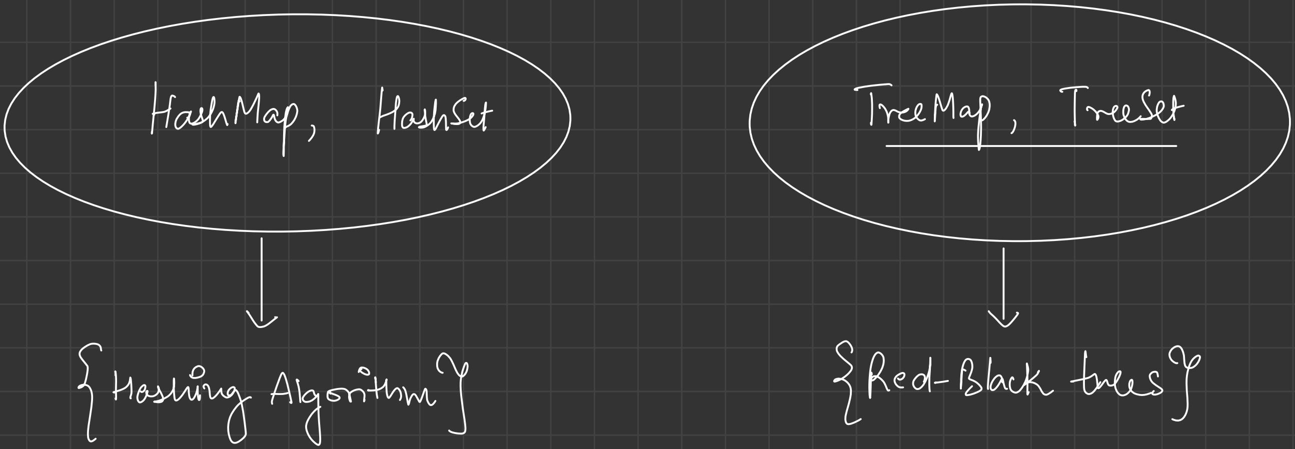
0	
1	
2	
3	3
4	4
5	13
6	
7	
8	
9	

Quadric Probing

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

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

$$f(i) = i^2, \quad \underline{\underline{0, 1, 2 \ldots}}$$



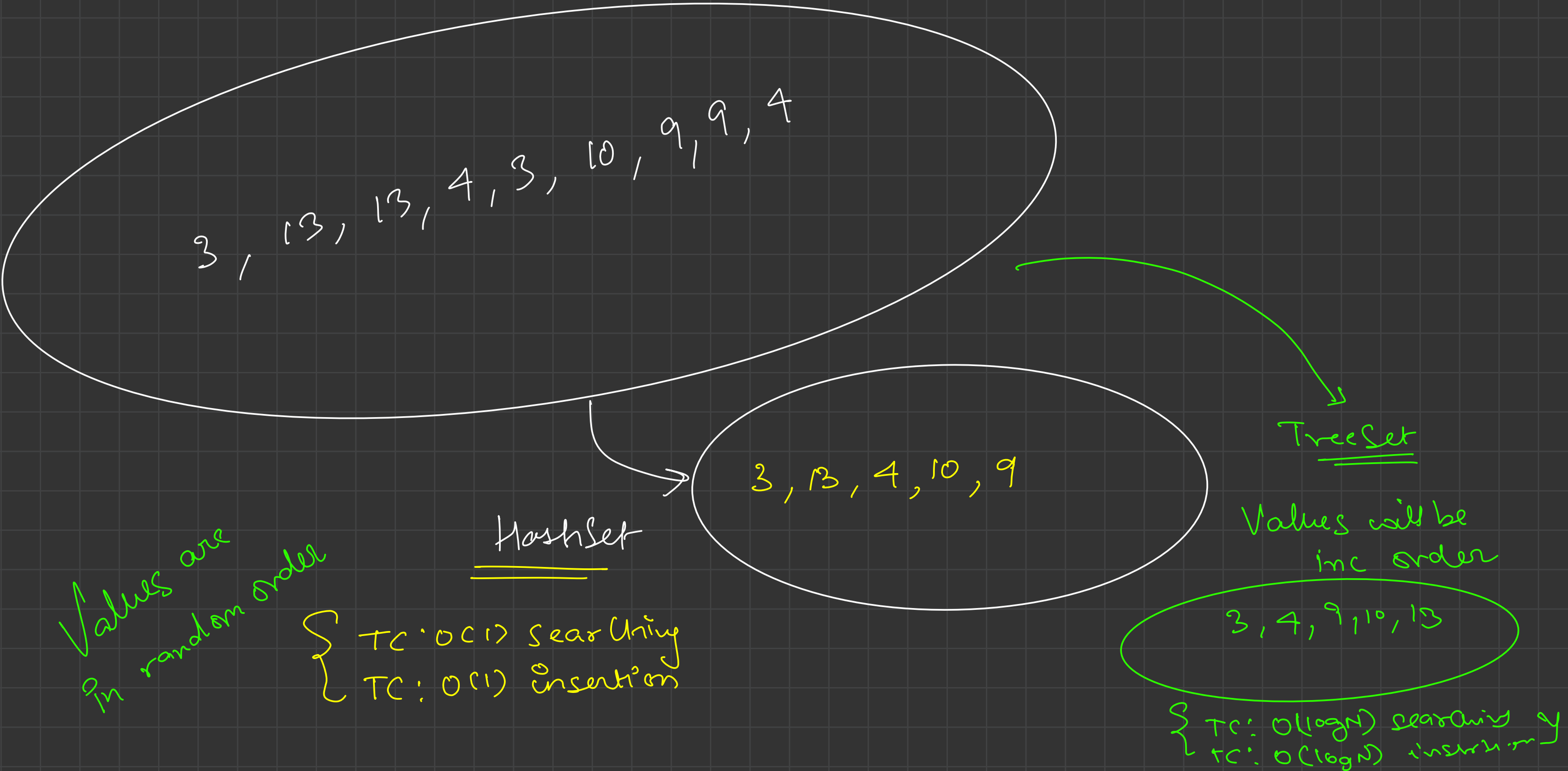
HashMap, HashSet

{Hashing Algorithm}

TreeMap, TreeSet

{Red-Black trees}

HashSet : collection of unique entity



3, 13, 13, 4, 3, 10, 9, 9, 4

HashSet

3, 13, 4, 10, 9

Values are in random order

{ TC : O(1) searching
TC : O(1) insertion

TreeSet

Values will be inc order

3, 4, 9, 10, 13

{ TC: O(logN) searching &
TC: O(logN) insertion

HashMap

→ key – value Pairs

HashMap< Integer, String > map = new HashMap();

1.	"Accio"
2.	"Pawan"
3.	"Anurag"

Key are stored in random order.

map.get(1) → "Accio"

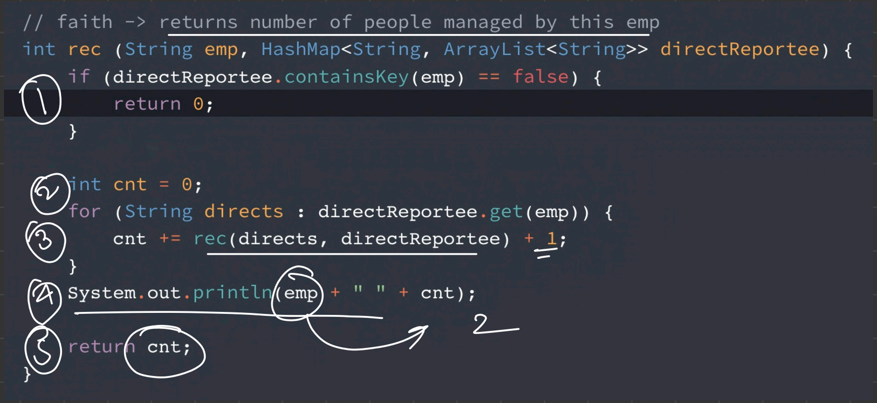
TC: O(1) → Searching

TreeMap

→ stores keys in asc. order

→ search TC: O(logN) , insertion TC: O(logN)

Break till 10'40pm

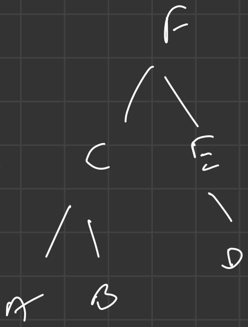


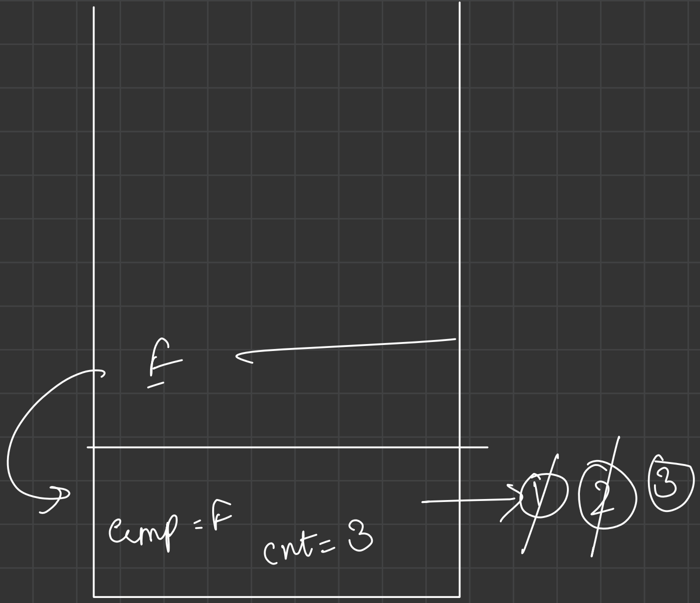
```
// faith -> returns number of people managed by this emp
int rec (String emp, HashMap<String, ArrayList<String>> directReportee) {
    if (directReportee.containsKey(emp) == false) {
        return 0;
    }

    int cnt = 0;
    for (String directs : directReportee.get(emp)) {
        cnt += rec(directs, directReportee) + 1;
    }
    System.out.println(emp + " " + cnt);

    return cnt;
}
```

{
F → {C, E}
C → {A, B}
E → {D}
}





Callstack