Hashing (Searching technique)

pool of Numbers { }

{ } int target

1. Linear Search TC: $O(N)$ SC: $O(1)$
2. Binary Search TC: $O(\log_2 N)$ SC: $O(1)$

} Store No. inside an array

3. Hashing TC: $O(1)$, SC: $O(1)$ } Storing No. inside a hashTable

NOTE: Hashing allows searching in TC: $O(1)$, SC: $O(1)$

if. 8, 3, 13, 16, 17, 14, 10 50

Boolean[] arr = { 0 1 2 3 4 5 50

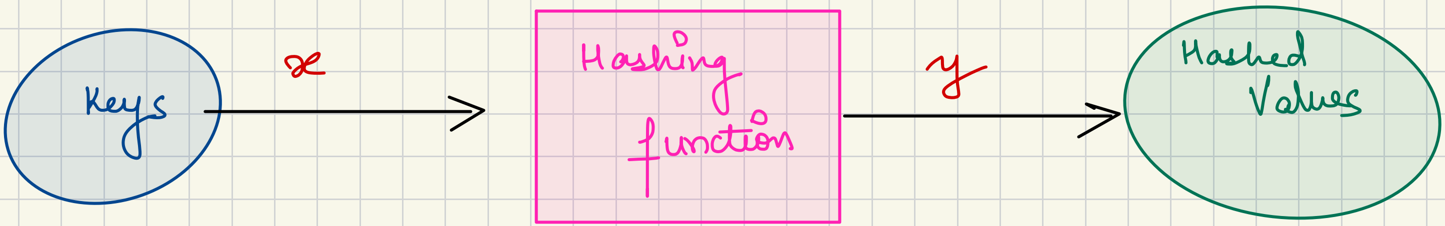
T T T T T T T . . . } . . T

```
search ( int key)        8
{
    if (arr [key] == true)
        return "found";
    else
        return "Not found";
}
```

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

- high Memory was consumed
- hence hashing was introduced

Basic Mechanism of Hashing:



Hash f^n: $\{$ 2-steps f^n $\}$

step1 $\quad g(x) = K$ $\quad$ integer

key

step2 $\quad h(K) = y$ $\quad$ hashed value

Key Space.

- ✓ 8
- ✓ 3
- ✓ 13
- ✓ 6
- ✓ 4
- ✓ 10

Hash f^n

$h(k) = K$ (one to one f^n)

$h(8) = 8$

$h(3) = 3$

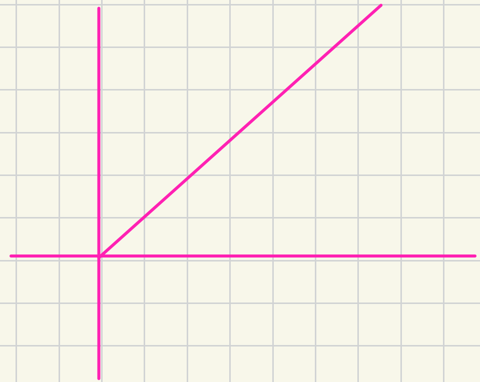
$h(13) = 13$

$h(6) = 6$

$h(4) = 4$

$h(10) = 10$

high memory was consumed



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	

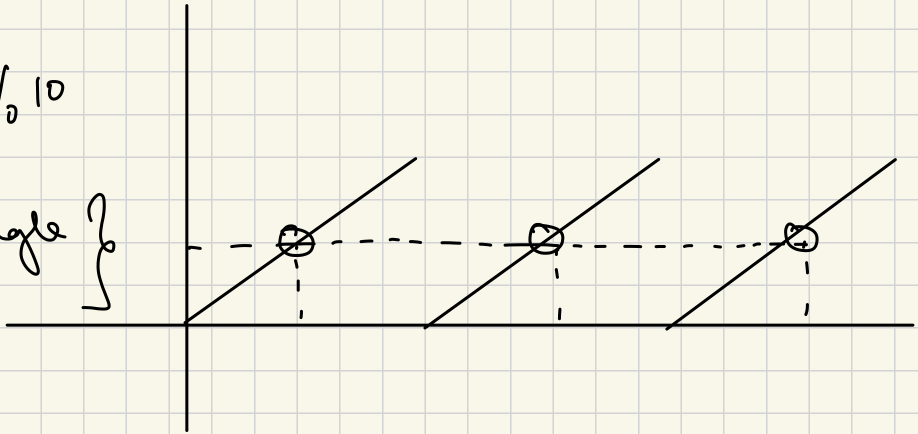
Pain Point

- unique hashed value for each key { one to one f^n }

many to one f^n

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

for multiple keys, there is single hashed values



Key Space.

✓ 8
✓ 3
✓ 16
13

Hash f^n

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

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

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

$$h(16) = 16 \% 10 = 6$$

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

Collision.

Hash Table

0	
1	
2	
3	3
4	
5	
6	16
7	
8	8
9	

Methods to remove collision

Open Hashing

1. Chaining

Closed Hashing

1. Linear Probing
2. Quadratic Probing

Chaining.

Key Space.

8
3
16
13
103

Hash f^n

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

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

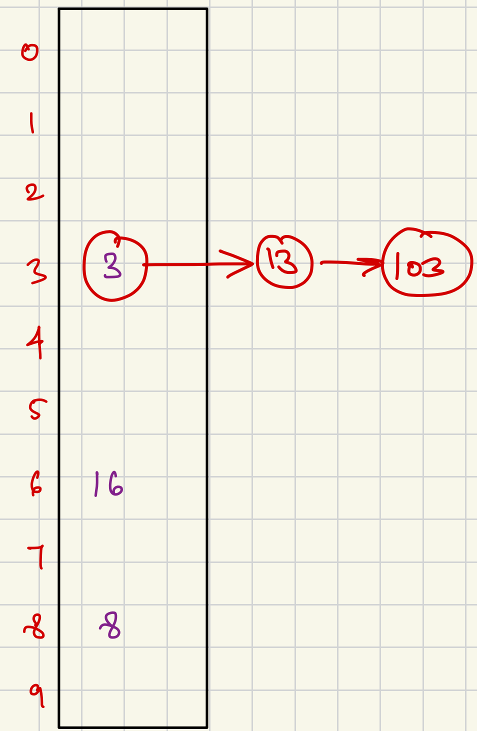
$h(3) = 3 \% 10 = 3$

$h(16) = 16 \% 10 = 6$

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

$h(103) = 103 \% 10 = 3$

Hash Table



Search (103)

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

Hash f^n

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

Load factor

0 - 10000

75% of Range

$\boxed{7500}$

Linear Probing.

Key Space.

- ✓ ~~8~~
- ✓ ~~3~~
- ✓ 16
- 13

Hash f^n

$$h'(K) = \{h(K) + g(i)\} \% 10$$

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

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

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

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

$$h'(16) = \{h(16) + g(0)\} \% 10 = (6+0) \% 10 = 6$$

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

$$h'(13) = \{h(13) + g(1)\} \% 10 = (3+1) \% 10 = 4$$

Hash Table

0	
1	
2	
3	3
4	13
5	
6	16
7	
8	8
9	

Quadratic Probing.

$$h'(k) = \{ h(k) + g(i) \} \% \ L.f.$$

$$h(k) = k \% \ L.f.$$

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

Based on hashing

1. HashSet
2. HashMap

TC: O(1) searching

TC: O(1) insertion

Red-Black trees

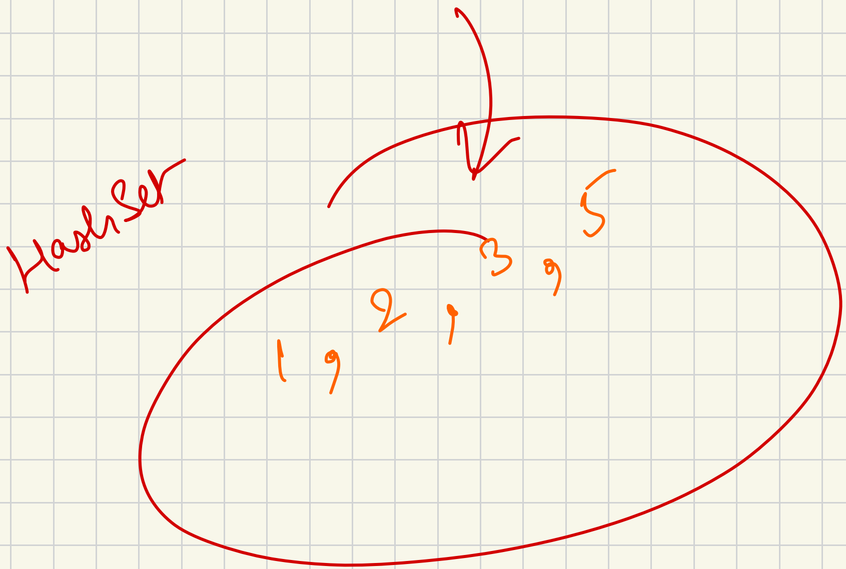
1. TreeSet
2. TreeMap

TC: $O(\log_2 N)$

HashSet

↳ set of unique entities

keySet = {1, 2, 3, 3, 5, 2, 1, 3, 5}



HashMap.

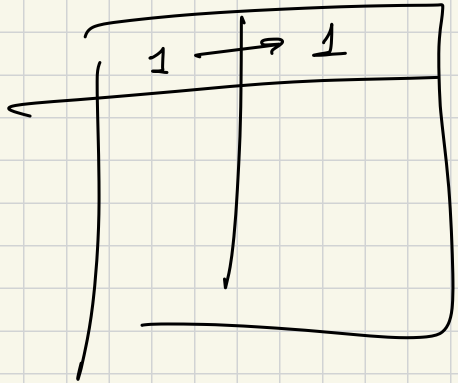
stores (Key, value) pair

implement shopping cart for E-commerce

String	Integer
Key (item)	value (qty)
(unique entity)	
Lays	~~4~~ 5
eggs	12
oranges	2
yogurt	4

Hashing is done on Keys

$\{1, 2, 3, 4, 4, 3\}$



```
10
~~203~~ ~~204~~ ~~205~~ ~~206~~ ~~207~~ ~~208~~ ~~203~~ ~~204~~ ~~205~~ ~~206~~
13
~~203~~ ~~204~~ ~~204~~ ~~205~~ ~~206~~ ~~207~~ ~~205~~ ~~208~~ ~~203~~ ~~206~~ 205 206 204
```

```
4
~~1~~ 1 ~~2~~ 5
4
~~1~~ ~~2~~ 3 4
```

3, 4, 1

H.W :

1. Valid Anagrams
2. Employees and Managers