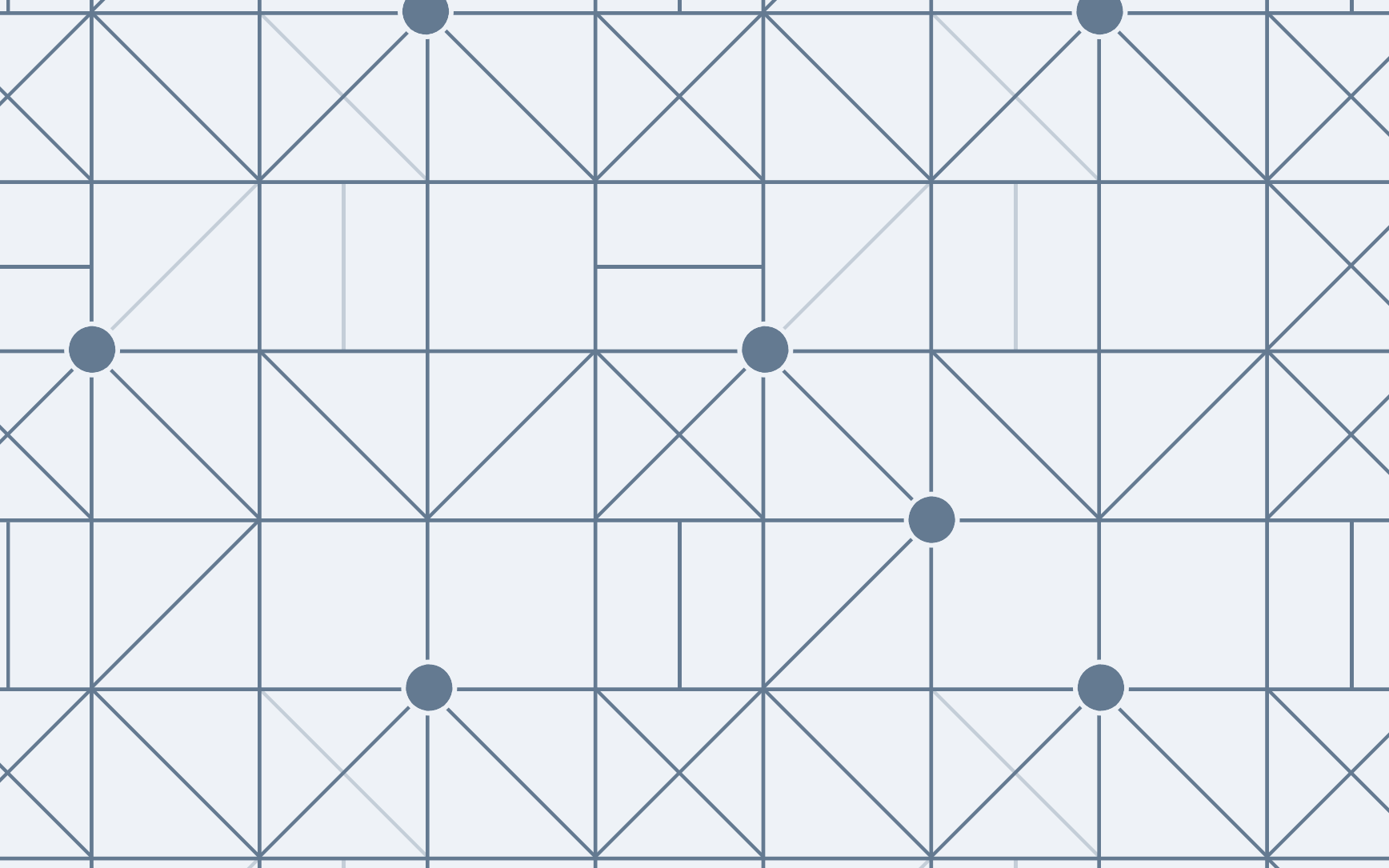


Hashing. {Theory}

Searching (TC: O(1))

A technique for searching purpose.

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

} {Storing ip in an array}

i/p 8, 3, 13, 16, 7, 4, 10 50

Hash Table

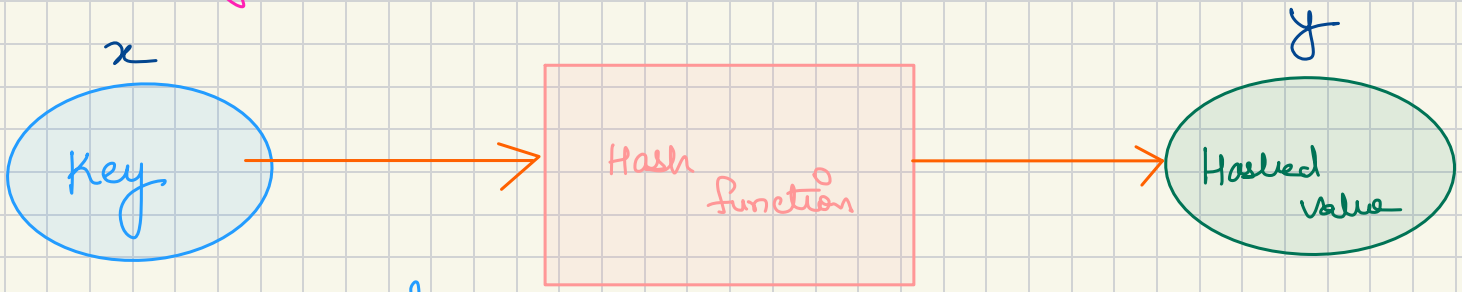
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	. . .	50
			T	T			T	T		T			T			T								T

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

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

high memory required, hashing was introduced to reduce this memory

Basic Hashing mechanism



{values inserted by users}

Key → can be anything

Hash fun^n {2-step fun^n}

step1 $g(x) = k$

Key → Integer value

step2 $f(k) = y$ → hashed value

Key Space

8
3
13
6
4
10
.
.

hash fun^n

$f(k) = k$ (one to one fun^n)

$f(8) = 8$

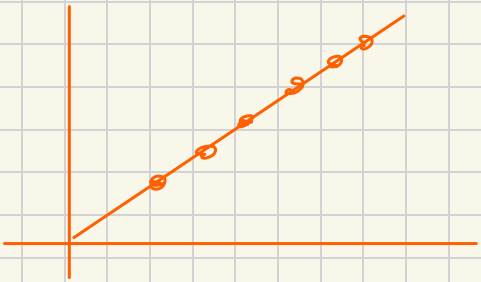
$f(3) = 3$

$f(13) = 13$

$f(6) = 6$

$f(4) = 4$

$f(10) = 10$



Hash Table

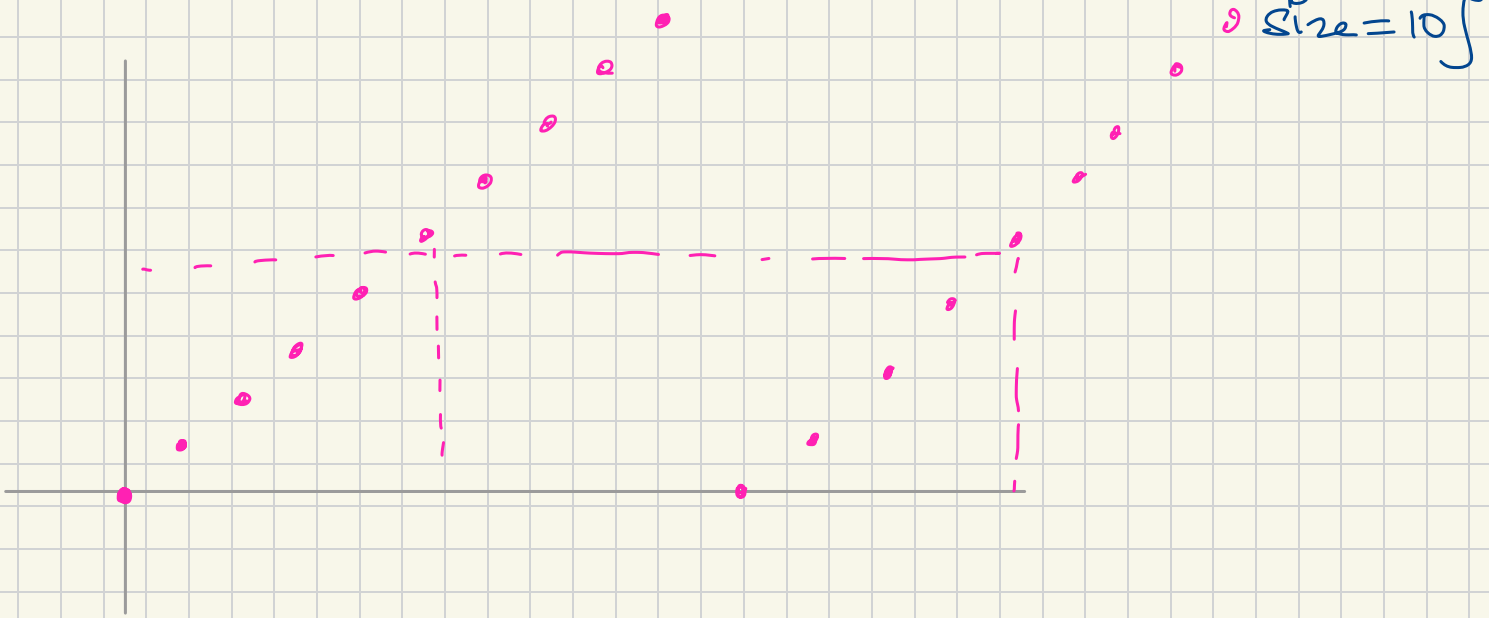
0	
1	
2	
3	3
4	4
5	
6	6
7	
8	8
9	
10	10
11	
12	
13	13
14	
15	
16	
17	
18	
19	
20	
21	

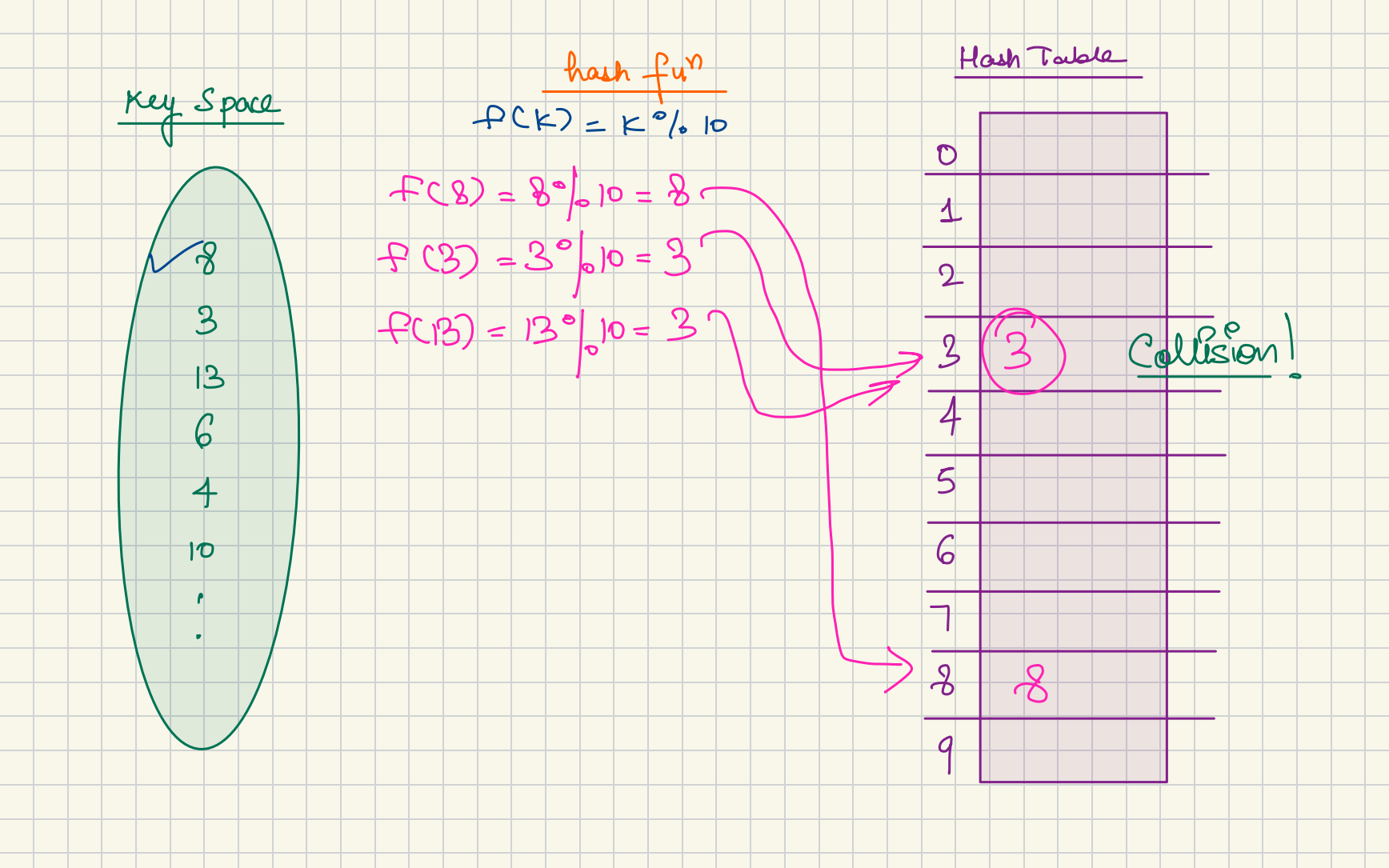
many to one relationship.

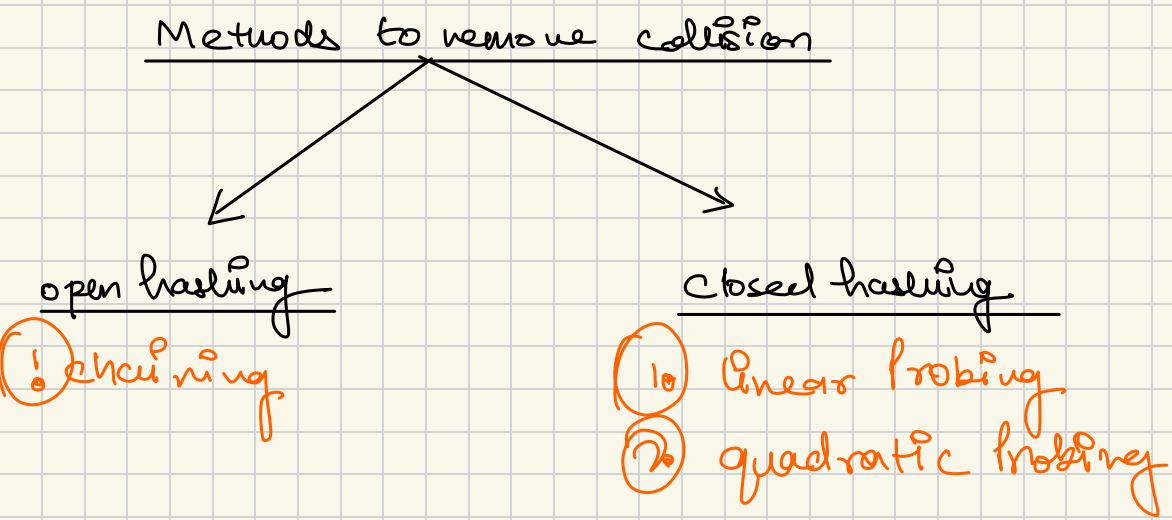
$$f(K) = K \% 10$$

hashed value : [0, 9]

size = 10




Key Space
8
3
13
6
4
10
hash fun
f(k) = k % 10
f(8) = 8 % 10 = 8
f(3) = 3 % 10 = 3
f(13) = 13 % 10 = 3
Hash Table
0
1
2
3
3
4
5
6
7
8
8
9
Collision!


Methods to remove collision
open hashing
1. chaining
closed hashing
1. Linear Probing
2. quadratic Probing

Chaining

Key Space

8
3
13
6
4
10
33
.

hash fun

$f(k) = k \% 10$

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

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

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

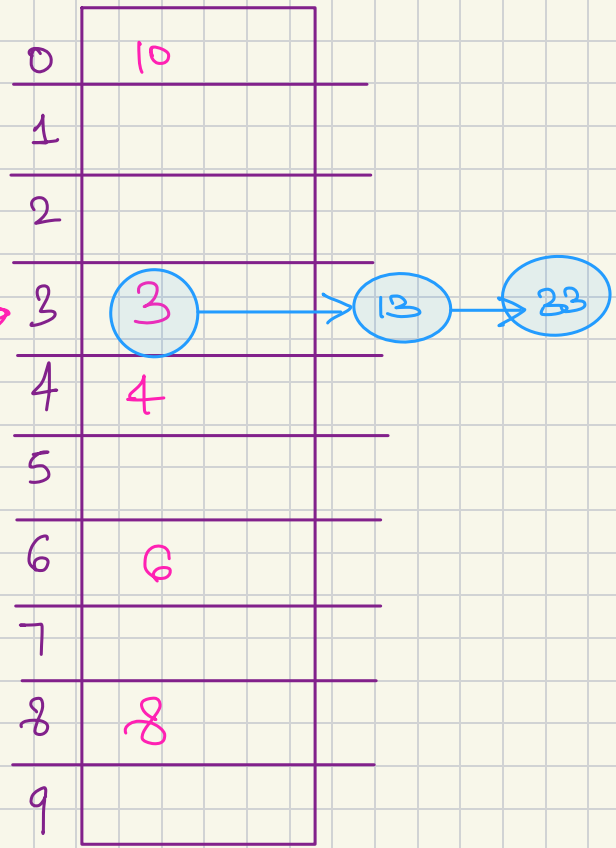
$f(6) = 6 \% 10 = 6$

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

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

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

Hash Table



searching(33) → hash value 3

avg TC : O(1)

Range of i/p: 0 – 100

size: 75% of Range

$$f(k) = k\% \text{ of } 75$$

Linear probing

Key Space

8
3
13
6
4
10
33
.

hash fun

$$f'(k) = \{f(k) + g(i)\} \% 10$$

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

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

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

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

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

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

searching(33)

→ hash value 3

avg TC : O(1)

Hash Table

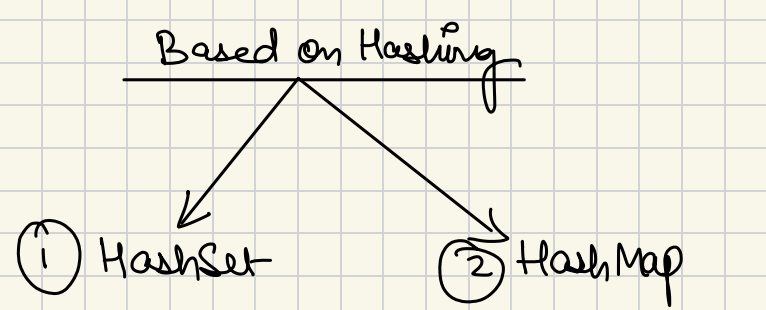
0	10
1	
2	
3	3
4	13
5	4
6	6
7	33
8	8
9	

Quadratic Probing

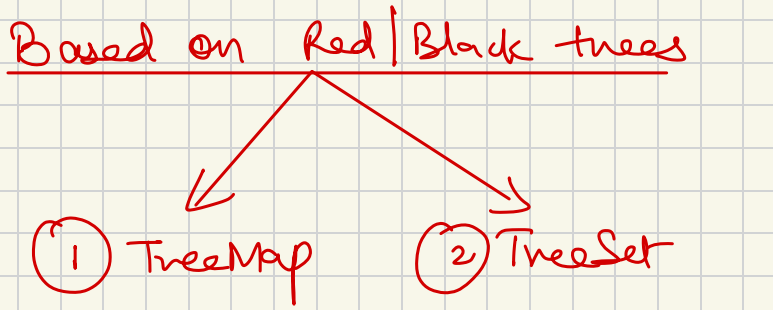
$$f'(k) = \{ f(k) + g(i) \} \% \text{ size}$$

where, $f(k) = k \% \text{ size}$

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



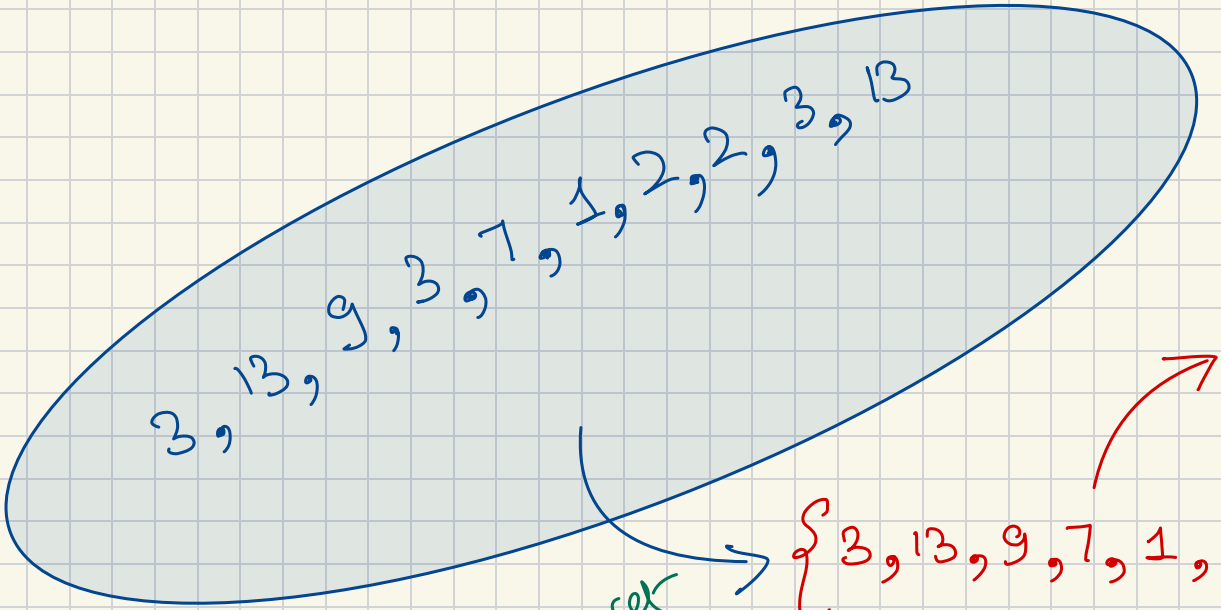
TC: O(1) ⇝ searching
TC: O(1) ⇝ Inserting



TC: O(log N) ⇝ searching
TC: O(log$_2$ N) ⇝ inserting

Hashset

↳ data structure to store unique values {entities}



Hashset → {3, 13, 9, 7, 1, 2}

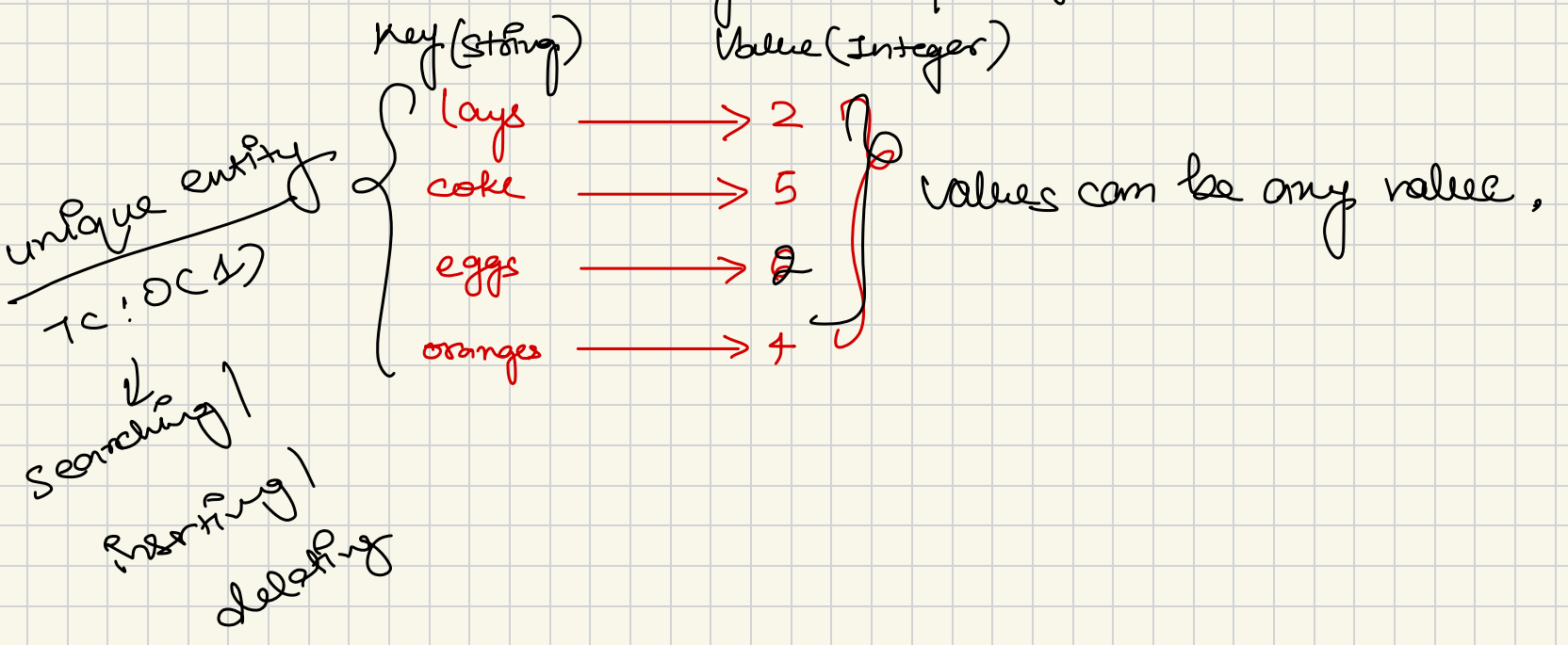
Random order

↳ searching TC: O(1)
↳ inserting TC: O(1)

HashMap.

↳ ds which stores key-value pairs!

shopping cart?

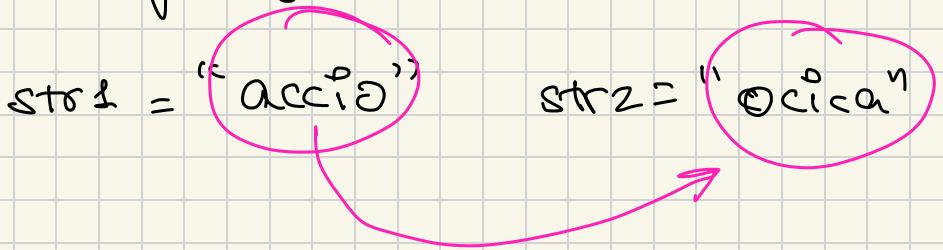


find All Duplicates in an Array

find the Ele, occ. K times

Valid Anagram

str1 = "accio" str2 = "ocica"



rearrangment

Brute force

Sorting

str1 = "accio" {Lexo}

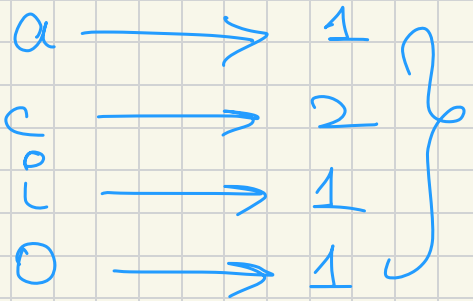
str2 = "accio" {Lexo}

Anagramic!

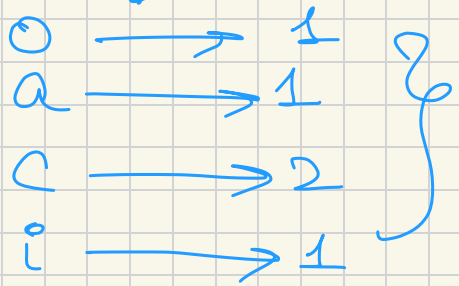
TC: O(NlogN)

SC: O(N)

str1 = "acciO"



str2 = "Ociac"



anagramic!