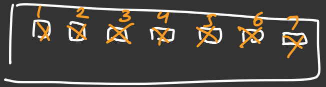
Hashing

Idea behind Hashing

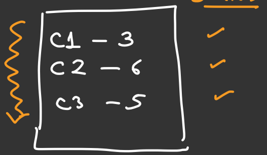
Hotel Manager



10 Rooms

Room is available or not.

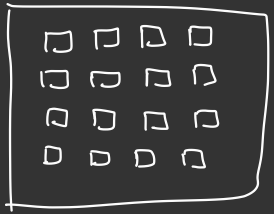
Register



① Scan the list whether the room ~~no~~ is available or not $\boxed{O(N)}$ for a small hotel ✓ N is less.

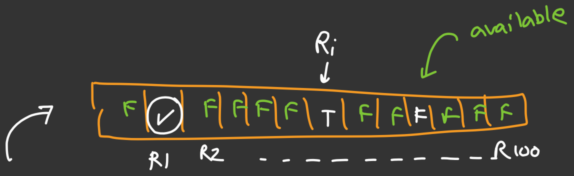
$\boxed{1 - 1000}$

Hotel 1000 Rooms



Online App

boolean array [1001]



$\boxed{\text{array[i] = true}}$ // Booking has been done

Check i^{th} Room is available or not ?

arr[i] == true ?   O(1) time   Search

New customers

→ arr[2] = true;   // inserting   O(1)

→ Customer checkout from room 2

arr[2] = false   // update   O(1)

Hashmap

↳ data-structure that allows to store data in the form key-value pairs

↳ allows operation like

- Insert (key, value)
- Search (key)
- Update (key, value)
- Delete (key)

Hotel Booking

Keys		Values
Room No		Status
3	–	True (Booked)
7	–	True (Booked)
6	–	False (Checked)

array index   array value

Relative ordering of items doesn't Matter

Go to Restaurent, Menu visit

Given a key, what is the value of key

Key → Item   Value → Price

Keys must be unique

Pizza - ₹ ~~100~~ ~~120~~ 180

~~Burger - ₹ 70~~ ✗

Drinks — ₹ 80

⋮

Array index can't be a String

```
→ Menu.insert ("Cheese Burger", 150)    // Insert
→ menu.find ("Maggi")                   // Search
→ menu.update ("Pizza", 120);
→ Menu.insert ("Pizza", 180);
→ menu.delete ("Burger");
```

O(1) time (for 99% of the cases)

Search ↓ Instant Reply | Jhatpat Search :)

O(1) time for most cases ↓ How → Later [Adv Batch]

Hashmaps are Unordered

Food Court

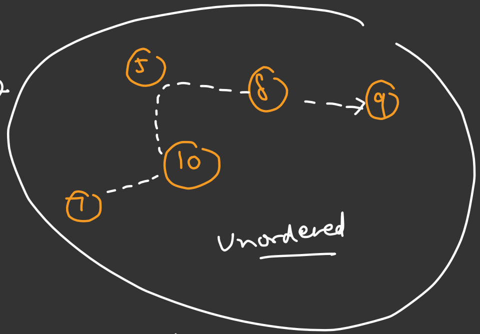
- Pizza
- ice-cream
- Shake

- ice-cream
- Pizza
- Shake

Random order

Insertion order

$$\underset{\#\ \text{Keys}}{\xrightarrow{5,\ 7,\ 8,\ 9,\ 10}}$$



Hashmap (object → heap)

$$\underbrace{9,\ 10,\ 5,\ 8,\ 9}_{\text{Random order}}$$

(Storage)

Drawback. → Not suitable for problems where ordering of data is important

$$\boxed{5 \mid 7 \mid 8 \mid 9 \mid 10}$$

Array → maintain Ordering

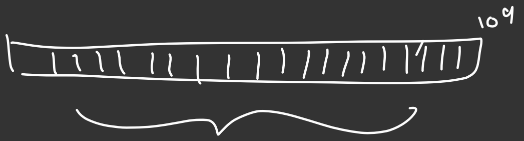
Int keys → can I always a use a arrays.

Hotel ⇒ Programmer

$\underbrace{1 \text{ to } 10^9 \text{ Rooms NO}}$

$\boxed{1000 \text{ Rooms}}$

$\underbrace{236, \quad 10059, \quad 5126, \quad 134 \ldots\ldots}_{1000 \text{ Rooms}}$



10^3 elements will be used

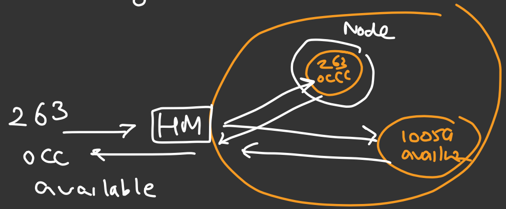
→ $\underbrace{10^9 - 10^3}$ washing these many Rooms ↓ take GBs

Hashmap

	K Room	V Status
hm <	236 ,	occupied >
hm <	10059 ,	available >
	⋮	⋮

<int, boolean>

<int, String>



Data

Library Version

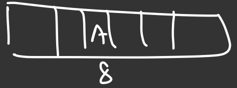
Build from Scratch in adv batch → Better understanding

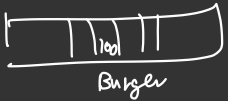
	Java	C++	Python	JS	C#
Both (K,V) ⇒	Hashmap	Unordered-maps	Dict	Map	Dictionary
(K) →	Hashset	Unordered-set	Set	Set	Hashset

Hashtable (synchronised later)

O(N) time

items = [burger, pizza, -----]
prices = [100 , 80, ----]





a[burger] = 100 ✗

O(1) → hm [burger] = 100 ✓

Booked Rooms → (101, 105, 108, 255)

is 255

hashset → O(1) search which is not provided by array or arralist

Store a list of Keys → insert, delete, update, search in O(1)

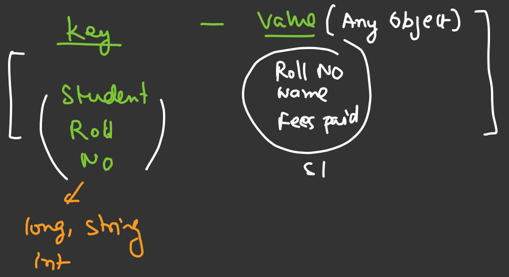
└→ in a" Unordered " manner

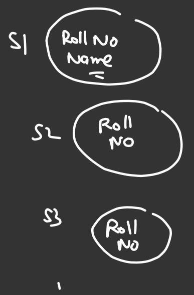
What Types of Keys

Default Implementation in Java Supports

↳ Primitive Type (int, double, ... char, boolean)

↳ String Type (string)





Later

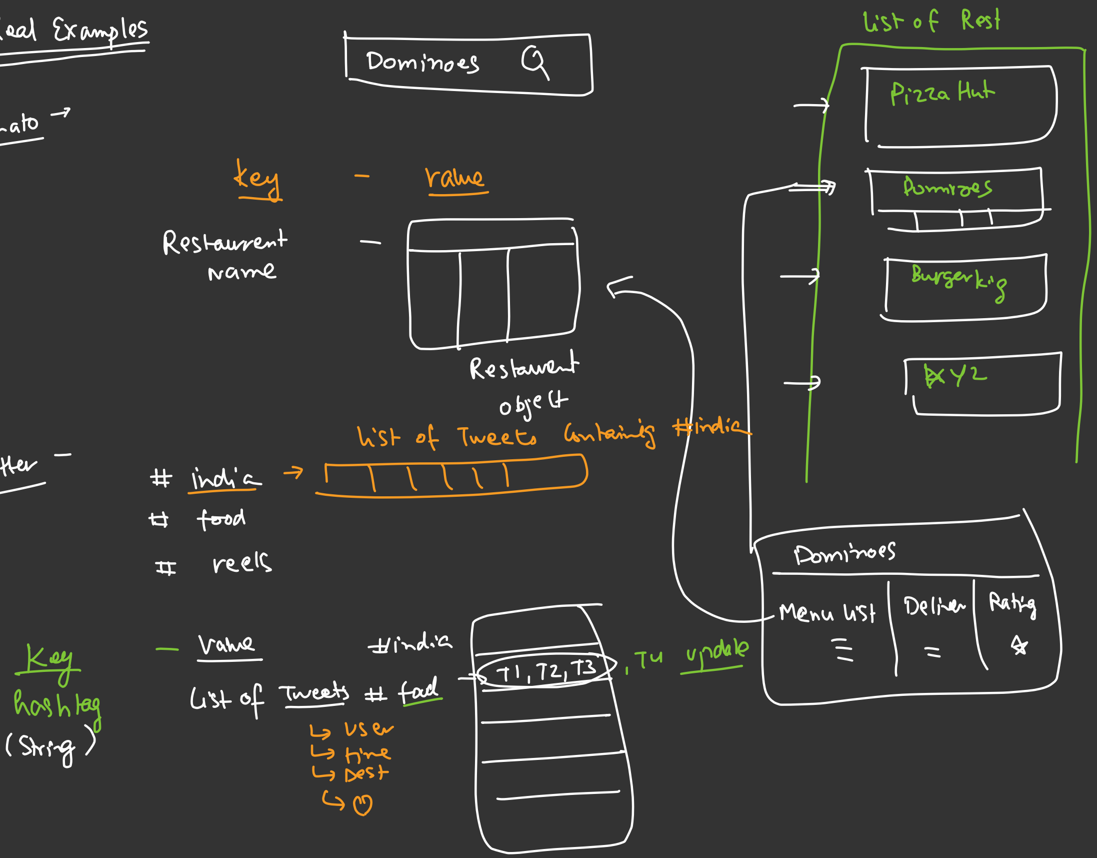
Custom hash fn

A way to Convert into some NO

Real Examples

Zomato →

Twitter –



hashmap< String, List<Tweet> >

↑

Q Store & query population of all country ?

hashmap< key , value >

key ↓ String (country name) or id

value ↓ long (population)

Q No of States in every country

Cnt ↓ No of States

Key → Country Name / string
Value → int / No of states

Hashmap <String, int >

Q Name of states in every country

India → (Punjab, H --
J&K --

key → country Name (String)
value → List of States

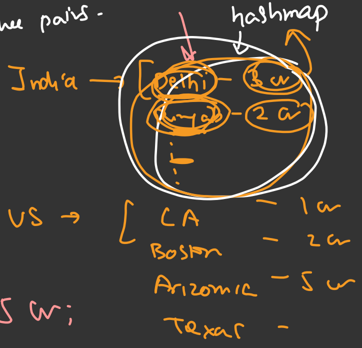
hashmap < String , Arraylist <String> >

Q For every Country, Store the population of each State

value is hashmap object

hashmap< String , hashmap<String, int> >

list of key-value pairs.

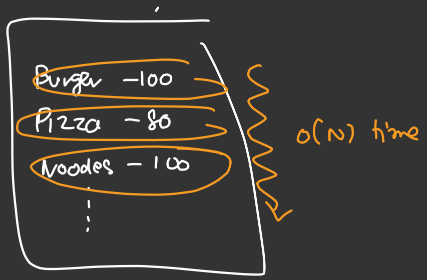


hm ["India"]["Delhi"] = 5 cr;

hm ["India"]. delete ("Delhi");

Q Key → value

↳ Give me something for 100 !
Burger , Noodles



Q Contact list (on your smart phone

Key - value

• Ajay - 9914316 2 f

↑ String name

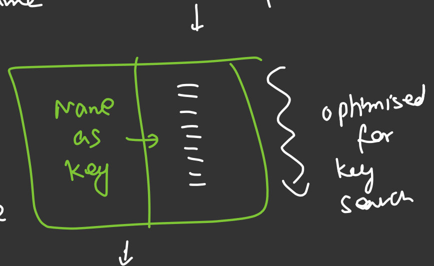
↑ String phone no

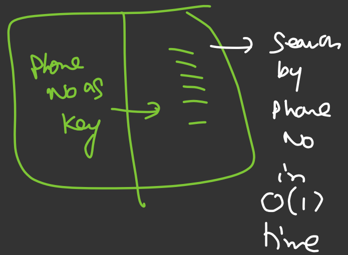
Phone NO → Name

9914316 2 f → MoM

Two hashmaps

Search → by name in O(1) time





↓

Search by value will
take linear time

String, List<String>

$$\text{burger} \rightarrow \underbrace{\begin{bmatrix} S, & M, & L \\ 50, & 100, & 150 \end{bmatrix}}_{\text{List of Int}}$$

$$\text{pizza} \rightarrow \underbrace{[200]}$$

↑ Key

↑ List<int>

Break :)

10.35

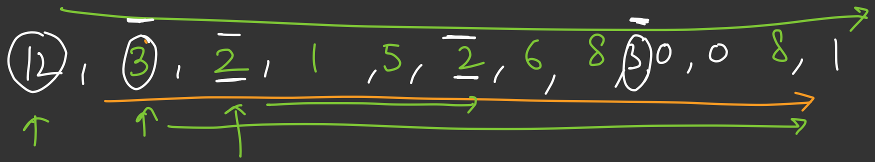
Key-String, value-hashmap

hashmap

Burger → { Aloo Tikki = 50, CheeseBurger - 80, King whooper - 200}

Pizza → { Paneer Onion - 95, ~~cheese~~ burst -

Q Given an array, find out the first (with least index) repeating element



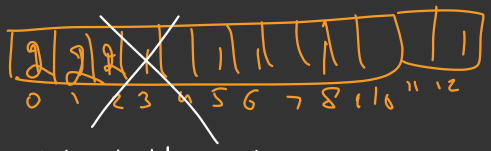
output → 2

I Brute force →

2 loops

a(i) Repeating

→ $O(N^2)$ time
→ $O(1)$ space



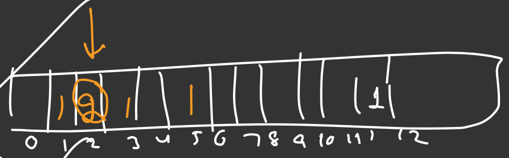
2

→ least i'dx ✗
least element ✓

II We can build a freq array

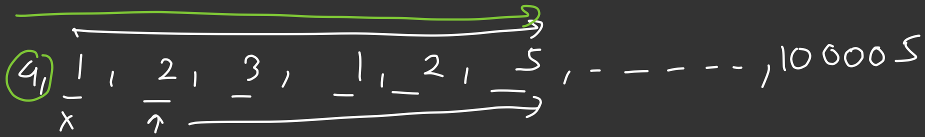
↳ the first time you update any freq as 2 → stop

↳ 2 repeating



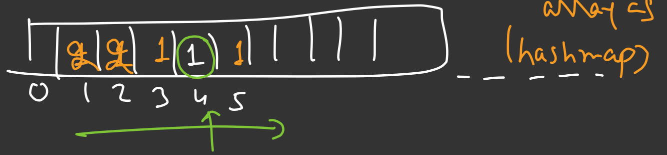
fails

Q First Non-Repeating element



Ap-1 ① Two loops $O(N^2)$

Ap-2 ②



✓ i) Build a freq table

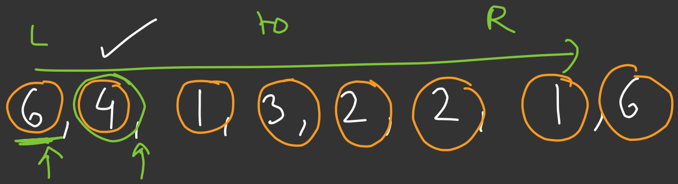
ii) iterate over the array again

Non-repeating 4

→ you will need a very big array $O(\text{Range})$

waste space

hash-table



First → least index

① Non-Repeating → freq 1

Key	—	value
element		freq
6	—	2
1	—	2
3	—	1
2	—	2
4	—	1

Random order

find the first one which is non-repeating

① Build a hash table → O(N) → O(N)

```
② for(i=0 ——— i<=N-1){
       if(hm.(a(i)) has freq 1){    → O(1)
           return a(i);
       }
   }
```

O(N) time
O(N) space

Q Given a array, find the no of ~~unique~~ distinct element
↳ any freq

6, 6, 1, 3, 4, 4, 3, 7

hashmap → { 6 – 2, 1 – 1, 3 – 2, 4 – 2 }

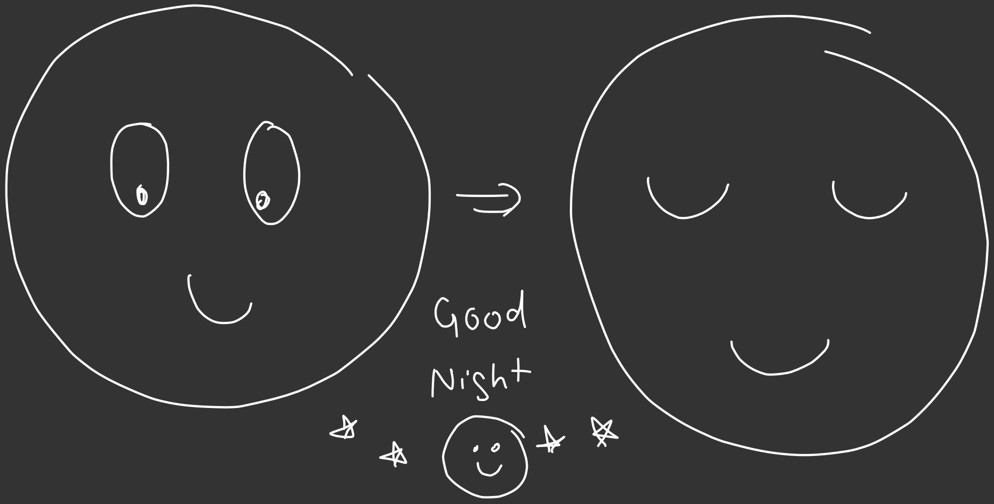
size → 4

hashset → { 6, 1, 3, 4, 7 } → No Duplicates are stored

size

unique keys

```
→ {
    hashset< integer > hs = new hashset<>()
    for(i=0, i<n ; i++){
        hs. insert( a[i] ),
    }
    return hs. size() .
}
```


Good
Night