1) Why Hashing?

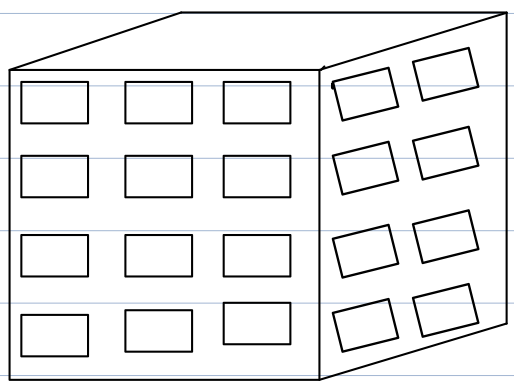
2) How is hashmap implemented? → Linked list, Binary Trees (self BBST)

3) Problems on Hashing

Volunteers

Lakshminadh Naidu

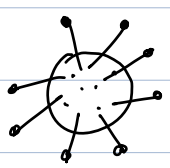
HOTEL (1000 rooms)



Register

Room No.	Status
1	A
2	0
3	0
4	A
.	
.	
.	
1000	A

Dec 2019



Numerologist

$[1 \quad 10^9]$

1000

[38, 19, 1, 1000, 10000, 1900, ...]

Pre-Covid Times

Array

size ⇒ 1000

Availability

```
bool arr[1000];
```

arr[i-1] → 0 → Not available

arr[i-1] → 1 → Available

Advantage of using arrays?

1) Lookup
2) Update

→ O(1)

Post Covid Times

size → 10^9

storing : 1000

```
bool arr[10^9]
```

bool → 1 Byte

1 → 1 Byte

10^9 → 10^9 Bytes ⇓ 1 GB.

Using

1000 → 1000 Bytes

Space is the wastage.

Requirements

1. No wastage of space
2. Fast Lookup — O(1)
3. Fast Update — O(1)

Hashmaps

very good Mugs.
↓
Breaks after having 5 cups of coffee

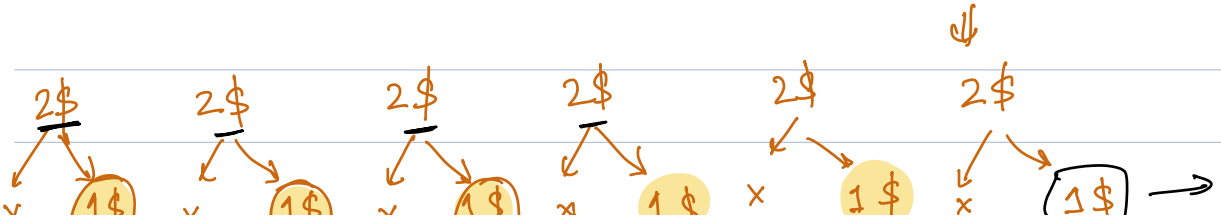
Free

1 $ 1 $ 1 $ 1 $ 1 $ 6 $ 1 $ 1 $ 1 $ 1 $ 1 $ 6 $

2 $ 2 $ 2 $ 2 $ 2 $ 2 $

Worst case : 6 $

Amortized Analysis



Amortized analysis : usually every operation is cheap & you have expensive operation once in a while.

What is a Hashmap?

unordered_map.

<Key, value>

Room No. → int

Availability → bool

int
float
string
double
char
bool

Hashmap <int, bool>

Application

Quiz population of every country.

Hashmap < Key , value >

country Name → string

population → int / long

Quiz all states of a country.

Hashmap < key , value >

name of country → all states → List<string>

Hashmap < string, List<string> >

For every country, store population of each state

<Key, value>

Key ⇓ country

value ⇓ population of each state

<Key, value>

Key ⇓ state name

value ⇓ population

Hashmap < string, Hashmap < string, int > >

Observation :

Value can be anything

Key can only be primitive data types
[int, float, double, bool, char], string

Can there be duplicate keys?

– NO!

HashSet	Set	unordered_set	Set	HashSet
Java	Python	C++	JS / Ruby	C#

Hashmap Dictionary unordered_map Map Hash table

Functionalities of Hashmap → O(1)

1. insert (key, value) // inserts into the hashmap (set)
2. get (key) // returns the value this key stores.
3. delete (key) // deletes the entire pair (set)
3. update (key, value)
4. size() // returns the no. of keys (set)
5. isPresent / contains / find // tells whether some key is present or not (set)

How to iterate over a map → Iterator. ⇒ O(#keys)

Qu 1. Given an array of size N & Q queries
Every query → integer x
Return the frequency of x

A: {2, 6, 3, 8, 2, 8, 2, 3, 8}

Queries: 10

2 ⇒ 3

8 ⇒ 3

6 ⇒ 1

.

.

Brute Force

For every query, iterate on the array to find the frequency

One query ⇒ O(N)

Q queries ⇒ O(N * Q)

Total TC O(N * Q)

Sort the array ??

TC O(N log N) + O(N * Q)

A: {2, 6, 3, 8, 2, 8, 2, 3, 8}

Hashmap

Step1: Iterate over array & store frequency of each element. } O(N)

2 : 3

6 : 1

3 : 2

8 : 3

Step 2: For each } Q queries

query, return the answer using hashmap | O(Q)

1 query → O(1)
Q → O(Q)

TC → O(N) + O(Q)
SC → O(N)

Code

```
Hashmap<int, int> mp

for(i=0; i<N; i++){
    if (A[i] is present in mp){
        update (A[i], mp[A[i]] + 1)
                //O(1)
    }
    else{
        insert (A[i], 1) //O(1)
    }
}

for(i=1; i<=Q; i++){
    Read x
    return mp[x]    // O(1)
}
```

mp[key]

O(N)

O(Q)

TC → O(Q+N)

Qu. Given an array, count the no. of distinct ele in the array.

A : {7, 3, 2, 1, 3, 7, 0}

Ans → 5

A: {6, 3, 7, 3, 8, 6, 9}

Hashmap < Key, value >

A[i] — int

frequency — int

① Create Hashmap.

② Return the size of the Hashmap

key	value
6	2
3	2
7	1
8	1
9	1

TC → O(N)

SC → 2xN

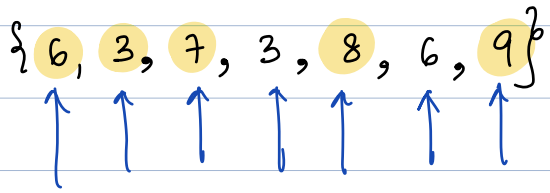
O(N)

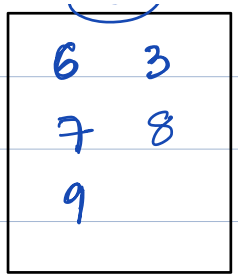
Introduce Hash set

↓

Stores unique keys.

(set)





Return the size of the set

TC → O(N)

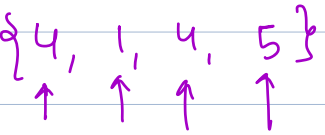
SC → O(N)

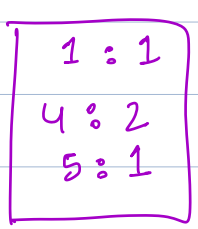
Code

```
HashSet<int> s;
for( i=0; i<N; i++ ) {
    insert ( A[i] into set s)
}
return s.size()
```

Break : 9:50

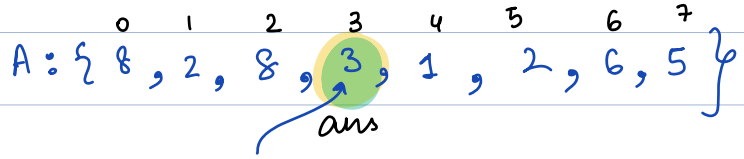
Unordered → order is not preserved of insertion

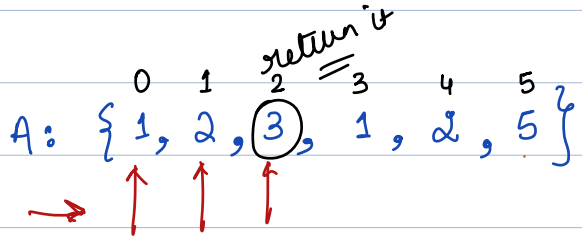




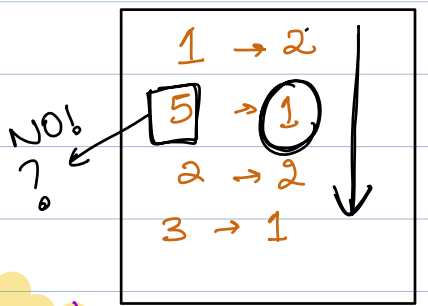
LinkedIn | Amazon

Que. Given arr of size N, find the first non repeating ele in the array.





HashMap (order is not maintain)



TC → O(N) + O(N)
= O(N)

SC → O(N)

For languages that maintains order of insertion in map, can iterate over map to get the answer.

Ordered Hashmap

- order of insertion
- sorted Hashmap based on keys

[6, 1, 2, 6, 1, 5, 2, 6]

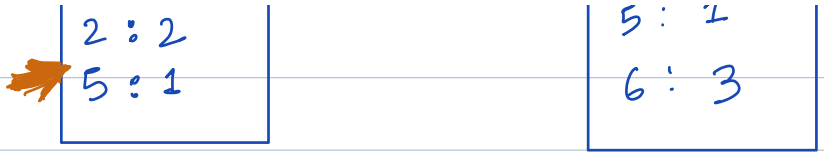
order of insertion.

6 : 3
1 : 2
↓

sorted based on key

1 : 2
2 : 2
↓

Java



Directi | Amazon | Myntra | Flipkart

Que. Given array, check if there exists a subarray with sum 0.

0	1	2	3	4	5	6	7	8
2	2	1	-3	4	3	1	-2	-3

Return true if a subarray with sum 0 is present.

Brute Force

= Generate all the possible subarrays.

```
for(i=0; i<N; i++) {
    for(j=i; j<N; j++) {
        sum = 0
        for(k=i; k<=j; k++) {
            sum += A[k]
        }
        if(sum == 0) return true
    }
}
return false
```

TC → O(N^3)

sum(i, j) = PS[j] − PS[i−1] // O(1)

create PS[]

```
for(i=0; i<N; i++) {
    for(j=i; j<N; j++) {
        sum = PS[j] - PS[i-1]
        if (sum == 0) {
            return true
        }
    }
}
return false
```

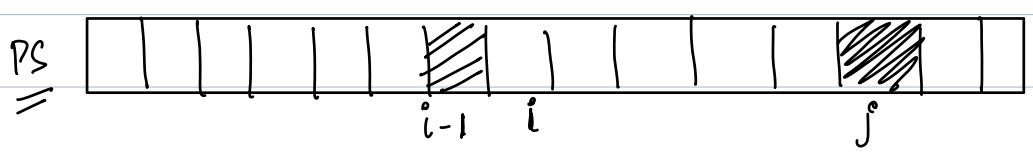
TC = O(N²)

Observation :

sum(i, j) = PS[j] − PS[i−1]

0 = PS[j] − PS[i−1]

PS[j] = PS[i−1]



PS[j] = sum(0, j)

$= \underbrace{sum(0, i-1)}_{PS[i-1]} + sum(i, j)$

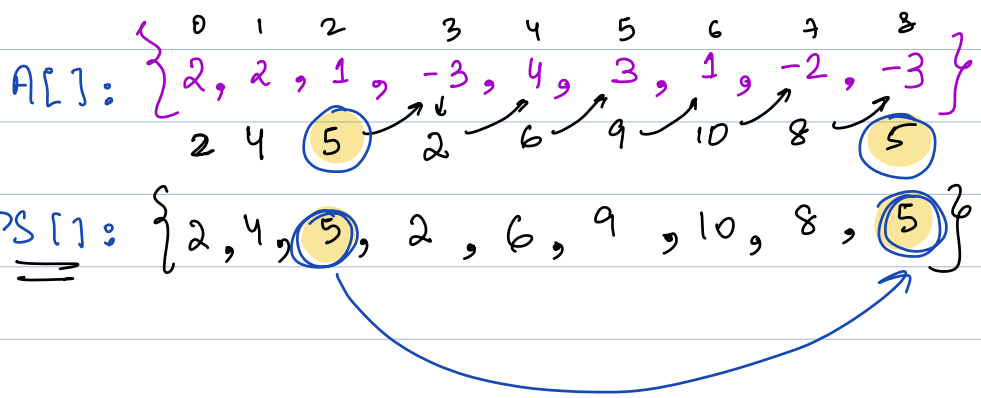
$PS[j] = PS[i-1] + sum(i, j)$

$sum(i, j) = 0$

Conclusion: If there exists a sum = 0 two values in PS array will be same

OR

If you find two repeating values in the PS array, then the sum b/w those 2 values = 0



Steps

1. Create a Prefix Array → O(N)
2. Check for duplicates → O(N)
   ↓ Set / Hashmap

+

$= O(N)$ TC

SC ⇒ O(N)

Edge Cases :

A : { 1, 2, 0, 8 }

PS : { 1, 3, 3, 8 }

A : { 0, 3, 2, 8 } ⇒ { 0 }, { 3 }, { 3,2 }, { 2 }, { 8 }, { 3,2,8 }, { 2,8 }, { 8 }

PS : { 0, 3, 5, 13 }

If one 0 is present in given array Return true

A : { 3, -1, -2, 4 }

PS : { 3, 2, 0, 4 }.

If PS has 0 return true.

Problem solving

MONDAY → 21st