Hashing 1

AGENDA:

- Introduction
- Questions

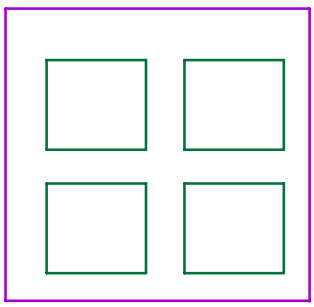
Rajat Sharma
Sarthak
Naval Oli
Vishal Mosa
Subhranil Kundu
Murali Mudigonda
Saurabh Ruikar
sharath r
Shrikanth
Vasanth
soumya hubballi
Rajendra
Akansh Nirmal
Shivansh Agarwal
amit khandelwal
Madhan Kumar M S
thulasi babu
Divyanshu
Pankaj
Sumit Adwani
PREETHAM
Gowtham Sankaran
Murali krishna Talluri
Vimal
Bhaveshkumar
Sanket Giri
Sahithi Kurmachalam
Venkata Sribhavana Nandiraju
suyash gupta
Sushant Patil
Shani
Sneha L
GAGAN KUMAR S
Rajeeb Kumar Nath
Sridhar Hissaria
Purusharth A
Shradha Srivastava
Nikhil Pandey
Abhishek Sharma

- freq of elements.
- First non repeating element.
- Count of distinct elements.
- Subarray sum 0

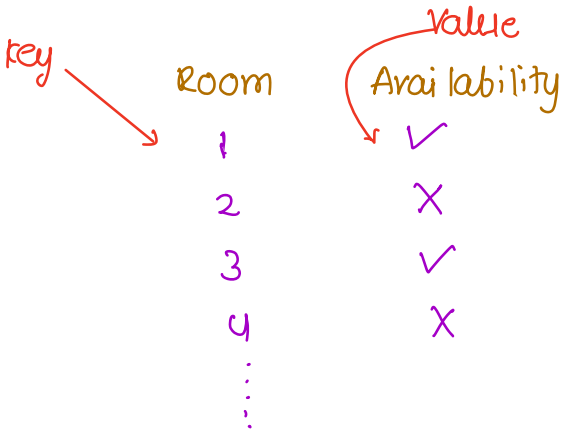
Current

~66 % ⟶ 70 %.

Shani owns ITC



HashMap



← Purushartha 3

Is it posible to have two rooms in a hotel with same no. ?

Therefore, there are no duplicate keys in a hashmap.

---

key in a hashmap must be unique

Value can be anything in the hashmap.

Store the population of countries in Hashmap ?

HashMap < String , Long >

String → Country name

Long → population

Store the no. of state in countries in Hashmap ?

HashMap < String , Integer >

String → country name

Integer → no. of states.

Store name of all states of every country.

name of all states → List<String>

country → key

HashMap < String , List<String> >

String → country name

List<String> → list of states.

Store population of each state in every country.

population of each state → HashMap < String , Long >

country → key

String → state

Long → population

HashMap < String , Long > → value

HashMap < String , HashMap < String , Long >>

Sometimes we just need to store the keys and not values.

HashSet → only stores keys. → keys will be unique.

{value = null}

HashMap & HashSet functionalities

HashMap

- insert (key, value) → will insert key, value
- size() → return size of hashmap
- delete (key) → deletes a key
- update (key, value)
- search (key) → checks for a key.

HashSet

- insert (key)
- size()
- delete (key)
- search (key)

TC : O(1)

	C++	Python	JS	C#
Hashmap	unordered_map	dict	map	Dictionary
HashSet	unordered_set	set	set	HashSet

A = [1 2 3]

```
hs // declare
for i ⟶ 0 to N-1 {
    hs.add(A[i])
}
```

The order of key in HS will be in no particular order.

3 1 2 , 3 2 1 1 2 3

Freq of elements

Given N elements and Q queries. Find the freq of the elements provided in a query

	0	1	2	3	4	5	6	7	8	9	10
A =	2	6	3	8	2	8	2	3	8	10	6

Q = 2 8 3 5

Ans	Q[i]	freq
	2	3
	8	3
	3	2
	5	0

Bruteforce — For every query Q[i], find the freq of Q[i] in A.

TC: O(Q * N) SC : O(1)

Create a freq array of size max(A)+1

A = 1 1 5 2 3 3

freq[] = 0 2 1 2 0 1

0 1 2 3 4 5

```
for i ⟶ 0 to N-1 {
|     freq[A[i]] ++
}
for i ⟶ 0 to Q-1 {
|     print (freq[Q[i]])
}
```

TC : $O(Q + N + \max(A))$ — $\max(A)$: freq init

SC : $O(\max(A))$ — 10^9

The above approach doesn't work for large A[i] values.

Use Hash Map

	0	1	2	3	4	5	6	7	8	9	10
A =	2	6	3	8	2	8	2	3	8	10	6

HM { 2 : ~~1~~ ~~2~~ 3
6 : ~~1~~ 2
3 : ~~1~~ 2
8 : ~~1~~ ~~2~~ 3
10 : 1

Pseudocode

```
HashMap<Integer, Integer> hm = new HashMap<>();
for (i ⟶ 0 to N-1) {
    // search for key in hm
    if (hm.containsKey(A[i])) {
        int freq = hm.get(A[i])
        hm.put(A[i], freq+1)
    }
    else {
        hm.put(A[i], 1)
    }
}

for (i ⟶ 0 to Q-1) {
    if (hm.containsKey(Q[i])) {
        print(hm.get(Q[i]))
    }
    else {
        print(0)
    }
}
```

TC: O(N+Q)

SC: O(N)

First Non Repeating element

Given N elements, find the first non repeating element.

	0	1	2	3	4	5	
A =	1	2	3	1	2	5	ans = 3

	0	1	2	3	4	5	6	7	
A =	4	3	3	2	5	6	4	5	ans = 2

	0	1	2	3	4	5	6	
A =	2	6	8	4	7	2	9	ans = 6

Approach

① → Create freq HM

② → Iterate over keys in HM and return the first key with freq == 1

keys in HM are unordered hence there is no guarantee of getting first unique

② Iterate over Array and check freq of A[i]==1

Pseudocode

```
HashMap<Integer, Integer> hm = new HashMap<>();
for (i ⟶ 0 to N-1) {
    // search for key in hm
    if (hm.containsKey(A[i])) {
        int freq = hm.get(A[i])
        hm.put(A[i], freq+1)
    }
    else {
        hm.put(A[i], 1)
    }
}

// Iterate over array

for (i ⟶ 0 to N-1) {
    if (hm.get(A[i]) == 1) {
        return A[i]
    }
}

return -1
```

TC : O(N)

SC : O(N)

Break : 22 : 45

Count of Distinct Elements

Given an array of N elements. Find the count of distinct elements.

A = 3 5 6 5 4 count = 4

A = 3 3 3 count = 1

A = 1 1 1 2 2 count = 2

Pseudocode

```
HashMap<Integer, Integer> hm = new HashMap<>();
for (i ⟶ 0 to N-1) {
    // search for key in hm
    if ( hm.containsKey ( A[i] )) {
        int freq = hm.get ( A[i] )
        hm.put ( A[i] , freq+1 )
    }
    else {
        hm.put ( A[i] , 1 )
    }
}
print (hm.size())
```

TC : O(N)

SC : O(N)

Use HashSet

```
HashSet<Integer> hs = new HashSet<>();

for (i ⟶ 0 to N-1) {
    hs.add(A[i])
}

print(hs.size())
```

TC : O(N)

SC : O(N)

Subarray sum 0 ****

Given an array of N elements. Check if there exits a subarray with a sum equal to 0.

	0	1	2	3	4	5	6	7	8	9
A =	2	2	1	-3	4	3	1	-2	-3	2

ans = true

Bruteforce

```
for i ⟶ N-1 {
    for j ⟶ [i, N-1] {
        total = 0
        for k ⟶ [i, j] {
            total += A[k]
        }
        if (total == 0) { return true }
    }
}
return false
```

TC : $O(N^3)$

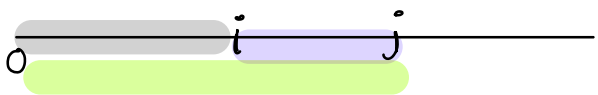
SC : $O(1)$

We either carry forward or prefix sum to reduce

TC ⟶ $O(N^2)$

		0	1	2	3	4	5	6	7	8	9
A	=	2	2	1	-3	4	3	1	-2	-3	2
PS	=	2	4	5	2	6	9	10	8	5	7

sum (i to j) == 0



sum (0 to j) − sum (0 to i−1)

PS[j] − P[i−1] = 0

PS[j] = ps[i−1]

Prefix sum at two different indexes j, i−1 are same.

if i == 0 then sum (0 to j) PS[j] == 0

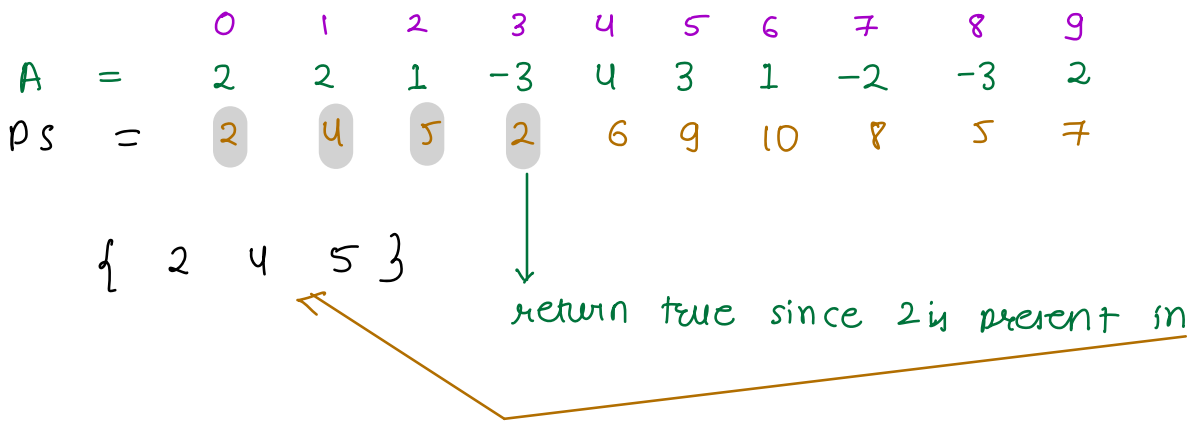
A	=	2	1	−3
PS	=	2	3	0

Conditions to check

① Repeated PS value

OR

② PS value == 0



Pseudocode

```
total = 0  // Running Prefix Sum.
hs // Hashset {Long} empty hashset

for i ⟶ 0 to N-1 {
    total += A[i]
    if (total == 0) return true
    // search for total in hs
    if (hs.contains(total)) return true
    hs.add(total)
}
return false
```

TC : O(N)

SC : O(N)

Doubt session

⟶ watch atleast 3-4 master classes on resume

⟶ Connect with a mentor for just resume