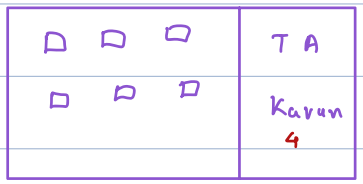
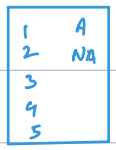
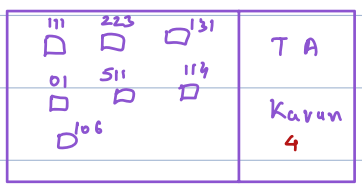
Hashmap Intro

Teekam Adarsh







10000

Room ⟶ Available or not Avail..

Key ⟶ value

Room number ⟶ T/F

Hashmap <Key, value>

Key has to be unique

Q1) Store population of every country

Key → Country Name : String
Value → population : long

Hashmap < String, long >

Q2) No. of states in each country

Key : Country Name : String
value : No. of states : int

Q3) For every country we want to know all states names

Key : Country Name : String
Value : List of all states : List <String>

Key

1) It has to be unique

2) Premitive data type [int, char, string, float, boolean, long]

1) Why unique ?
2) Why premitive ?
3) How does HM work ?

Advance

Hashing algo working ?

Value :

No restriction

Hash code Advance

HW

Q4) For every country store population of each state

Hashmap Hashset

Hashmap	Hashset
<key, value>	<key>
key will be unique	key will be unique
	Then cannot be any duplicate

{ Sunday , Moday .... }

HS: {1, 2, 3}

insert 2

Hashmap functionality

TC: O(1)

HM	HS
size(): No. of keys in HM	size: No. of keys in HS
void insert (K,V): Insert (K,V) in HM	insert (K) nset K in HS
bool search (K): check if K in HM	
DT for V get (K): value associated with key	search (K) check if K in HS
void delete (K): Del (K,V) in HM	delete (K)
void update (K, newV): update newV corresponding to key	

{ # Hashing library names in different languages }

pseudocode	Java	python	C++
Hashmap	Hashmap	dictionary	unordered_map
Hashset	Hashset	set	unordered_set

only for syntax : Use ChatGPT

Q1) Find Frequency of Numbers

Given N array elements & Q queries for each query find frequency of each element in array

$1 <= N <= 10^5$ $1 <= Q <= 10^5$

	0	1	2	3	4	5	6	7	8	9	10
A[10] = {	2	6	3	8	2	8	2	3	8	10	6 }

Q: freq

2 : 3

3 : 2

4 : 0

10 : 1

q[] :

TC: O(QN)

Brute force

Iterate on every Q

Iterate and count no. of occur.

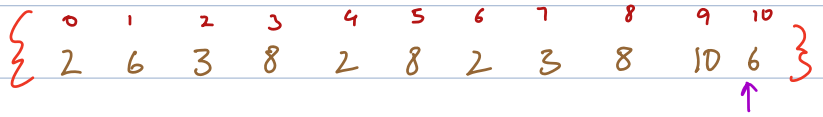
if (A[i] == q)

count++

HM

key → value

element freq



HM

2 : 3

6 : 2

3 : 2

8 : 3

10 : 1

step1) Construct the HM

```
hm = Hashmap()
for(i=0; i<N; i++){
    if (hm.search(A[i])){
        hm[A[i]] += 1
        // hm.update(A[i], hm.get(A[i])+1)
    }
    else{
        hm.insert(A[i], 1)
    }
}
```

TC: O(N+Q)

SC: O(N)

```
for(i=0; i<len(Q); i++){
    if (hm.search(Q[i]){
        print(hm.get(Q[i]))
    }
    else{
        print(0)
    }
}
```

1, 2, 3, 4

Microsoft

Q2) Find the first non-repeating element

5: 1
1: 2
2: 2
3: 1

A[6] = { 1 2 3 1 2 5 } ⟶ 3
R R NR

A[8] = { 4 3 3 2 5 6 4 5 } ⟶ 2
R R R NR

A[7] = { 2 6 8 4 7 2 9 } ⟶ 6
R NR

{ 1, 1, 2, 2 } ⟶ -1

1) Build the HM
2) Find the first key having value = 1

1) Build the HM

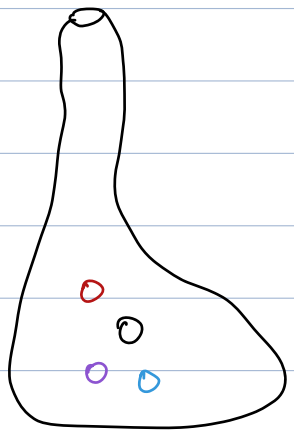
2) for(i=0; i<N; i++) {
    if (hm.get(A[i] == 1) {
        return (A[i])
    }
}

return -1

TC: O(N)
SC: O(N)

10:42

Hashmap is unordered



Pinkush : 2

Anirban : 1

Teekam : 4

Avinash : 3

(Q4) Given ar[N] elements find no. of distinct elements after removing duplicates.

A[5] = { 3 5 6 5 4 } ⟶ { 3 6 5 4 } ⟶ 4

A[5] = { 1 1 1 2 2 } ⟶ {1, 2} ⟶ 2

A[3] = { 3 3 3 } ⟶ {3} ⟶ 1

Use · hashset

```
for(i=0; i<N; i++){
    hs.insert(A[i])
}
```

return hs.size()

Google / Amazon / Uber

Q4) Given arr[N] elements, check if there exists a subarray with sum = ~~0~~ X HW

T F

	0	1	2	3	4	5	6	7	8	9
A[10] = {	2	2	1	-3	4	3	1	-2	-3	2 }

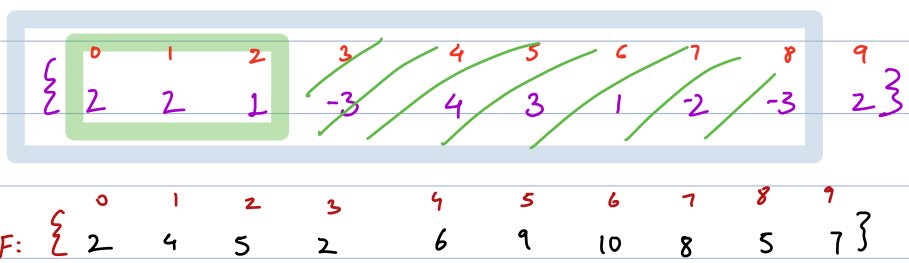
True

	0	1	2	3	4	5	6	7	8	9
A[10] = {	2	2	1	-3	4	3	1	-2	-3	2 }

1) Find sum of every subarray TC: $O(N^3)$

↓

Pf TC: $O(N^2)$



PF: {	0	1	2	3	4	5	6	7	8	9 }
	2	4	5	2	6	9	10	8	5	7

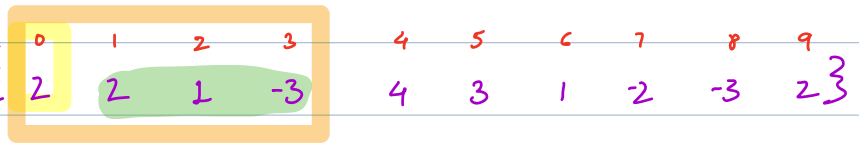
pf[2] = 5 : sum(A[0:2])

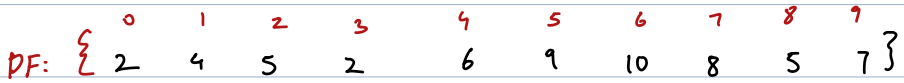
pf[8] = 5 sum(A[0:8])

sum(A[0:8]) = sum(A[0:2]) + sum(A[3:8])

5 = 5 + sum(A[3:8])

sum(A[3:8]) = 0





$$\text{sum}(A[0:3]) = \text{sum}(A[0:0]) + \text{sum}(A[1:3])$$

$$2 = 2 + \text{sum}(A[1:3])$$

$$\text{sum}(A[1:3]) = 0$$

TC: O(N)

SC: O(N)

1) Find pf sum array if pf[i] == 0 ? return true
2) check if there is repeatition or not ?

HM

freq > 1

{5 3 5 4} —HS→ {5 4 3}

{1 2 3 4} → {1, 2, 3, 4}

✓

```
for(i=0; i<N; i++){
    hs.insert(A[i])
}
if(hs.size() < N){
    return true
}
else{
    return false
}
```

A: [1 2 -3 5]

pf: [1 3 0 5]

pf[i] = 0

sum(A[0:i]) = 0

[3 5 6 0 1]

pf [3 8 14 14 15]

[0