Hashing Basics

Scenario1

↳ 1000 rooms labelled as : [1, 1000]

↳ occupied / not occupied

bool room[1001] ⇒ room[i] = true [if ith room is occupied]

since rooms are called from [1, 1000] NOT [0, 999]

⇒ room[i] = false [else]

Scenario 2

↳ 1000 rooms labelled between [1 to 10^9]

bool room [10^9+1]

↳ Issues : Huge space wastage

↳ Advantage : TC : O(1)

Hashmap stores <key, value> pair

<10015, occupied>

<123, unoccupied>

⋮

⇒ check in 10015 ? occupied : O(1)

TC : O(1) to search

SC : O(N) to store

N room entries

Note : Keys are unique

value can be anything

Hashmap< int, bool>

Question 1

Store population of every country

Key : country name → string

value : population → int / long

Hashmap< string, long> hm ⇒ pseudo syntax

Q2 No. of states of every country

Key → country name : string

value → count of states : int

Hashmap< string , int > hm

Q3 Name of all states of every country

Key : country name : string

value : all state names : list<string>

↳ C++ : vector

↳ py : list

↳ java : Arraylist

Hashmap< string, list<string>>

Q4 population of each state in every country

Key : country name → string

value : population of each state] → Hashmap < string , long >

↓ state name

↓ population

Hashmap < string , Hashmap < string , long > > hm

We observe 2 things :

1. Value can be anything

2. Key can only be primitive datatype.

↓

int / long / float / double / string / char

Hashset<key>

- → it only store keys
- → keys have to be unique
- → only primitive datatype

Hashmap functionality

size : { # keys present }

insert (key, value)

search(key)

delete (key)

update(key, value)

Hashmap

~~<India, 800>~~

<US, 200>

<India, 900>

override

Hashset functionality

size : { # keys present }

insert(key)

search(key)

delete (key)

All operations here are O(1)

→ Hashing libraries name in diff. languages

Pseudocode	Java	C++	Python	JS	C#
Hashmap	Hashmap	unordered_map	dict	map	dictionary
Hashset	Hashset	unordered_set	set	set	Hashset

Question 1

Given N array elements & Q queries.

for each query find freq. of given element in the array.

a[11] = {2 6 3 8 2 8 2 3 8 10 6}

Q: 4		freq
2	:	3
8	:	3
3	:	2
5	:	0

Constraints:

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

$1 <= a[i] <= 10^9$

Idea1: for every query iterate & get count

TC: O(Q×N)   SC: O(1)

Idea 2: Store data in hashmap

Key → array elements : int

value → freq. of element : int

Hashmap<int,int> hm

{2 6 3 8 2 8 2 3 8 10 6}

<2,3>   <8,3>
<6,2>   <10,1>
<3,2>

Code

```
Hashmap<int,int> hm

for(i=0; i<n; ++i) {

    if( hm.search(a[i]) == true ) {
        // a[i] is already present
        hm[a[i]]++   // update

}
```

TC: O(N)

```
        else {
            hm.insert( {a[i],1} )  //insert
        }
    }

for (i=0; i<Q; ++i) {                          TC:O(Q)
    if( hm.search( input[i]) = = true ) {
        print( hm[input[i]])
    }
    else{
        print(0)
    }
}
```

TC: O(N+Q)

SC: O(N)

Question 2

find the first non-repeating element

$a[6] = \{ \overset{\times}{1} \quad \overset{\times}{2} \quad 3 \quad 1 \quad 2 \quad 5 \}$ ans = 3

$a[8] = \{ \overset{\times}{4} \quad \overset{\times}{3} \quad \overset{\times}{3} \quad 2 \quad 5 \quad 6 \quad 4 \quad 5 \}$ ans = 2

Idea 1

1. Insert all elements in hashmap
2. Iterate over hashmap to get first key with value 1.

Note: Order of insertion of keys is not maintained in hashmap / hashset.

Idea 2 :

1. Insert all elements in hashmap → O(N)
2. Iterate over array & get first element with $m[a[i]] == 1$ → O(N)

TC : O(N) SC : O(N) ⇒ TODO

Question 3

Given N elements, find no. of distinct elements.

$a[5] : \{\overset{\checkmark}{3}\quad \overset{\checkmark}{5}\quad \overset{\checkmark}{6}\quad 5\quad \overset{\checkmark}{4}\}$ ans = 4

Idea

insert all elements in hashset

$a[7] = \{6\ 3\ 7\ 3\ 8\ 6\ 9\}$

HashSet<int> hs ⇒ hs.size = 5

$\boxed{6, 3, 7, 8, 9}$

Note: In hashset, if same key is inserted multiple times, we will store only 1 occurance

Code

```
Hashset<int> hs
for(i=0; i<n; ++i){
    hs.insert(a[i])
}
print(hs.size())
```

TC : O(N)

SC : O(N)

Question4

Given N elements, check if there exists a subarray with sum=0

	0	1	2	3	4	5	6	7	8	9
a[10] =	2	2	1	-3	4	3	1	-2	-3	2

ans = true

Idea: for every subarray, calculate sum

- $O(N^3)$ nested loops
- $O(N^2)$ prefix sum
- $O(N^2)$ carry forward

TC: $O(N^2)$
SC: $O(1)$

	0	1	2	3	4	5	6	7	8	9
a[10] =	2	2	1	-3	4	3	1	-2	-3	2
pf[10] =	2	4	5	2	6	9	10	8	5	7

Obs1: If pf[], numbers are repeating?

$pf[0] = 2 \quad = sum[0,0]$

$pf[3] = 2 \quad = sum[0,3] \quad = sum[0,0] + sum[1,3]$

$\cancel{x} \quad = \cancel{x} \quad + sum[1,3]$

$$\boxed{sum[1,3] = 0}$$

Doubt :

$a[4] = \{ 2 \quad -5 \quad 3 \quad 6 \}$

$pf[] = \{ 2 \quad -3 \quad 0 \quad 6 \}$

↳ in pf[] no repetition but subarray sum=0

$pf[2] = 0 \Rightarrow sum[0,2] = 0$

→ Note : In your pf[] every is single 0 is present, there exist a subarray with sum=0 final obs :

If ele repeat in pf[]
OR
If 0 is present in pf[]
→ there exist subarray with sum=0

Code

```
bool zerosum( int a[] , n) {
    pf[n]    // construct pf[] → TODO
    Hashset<int> hs
    for (i=0; i<n; ++i) {
        if(pf[i]==0) { return true; }
        hs.insert(pf[i])
    }
    if (hs.size() < N) {   //repeatition in pf[]
        return true
    }
    return false
}
```

TC: O(N)

SC: O(N)

Extra Question

Given N array elements, find count of subarrays with sum=0.

Since ans can be large print resut % (10^9+7)

$A = [1 \quad -1 \quad -2 \quad 2]$ ⇒ $[1 \quad -1]$

$pf[\,] = [1 \quad 0 \quad -2 \quad 0]$

$[-2 \quad 2]$

$[1 \quad -1 \quad -2 \quad 2]$