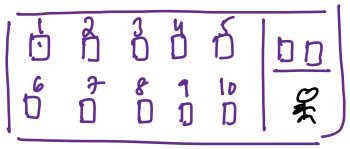
Todays Content:

7:05am Starting

- → Hashmap Intro
- → Frequency of each query
- → first non-repeating element
- → # distinct elements
- → # Subarray with sum = 0

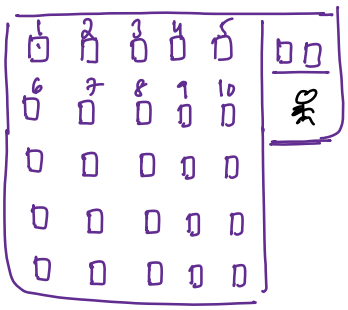
// Hashmap Intro:

a) Chandra Rakesh → Register



Room No	Availability
1	✗
2	✗
3	✓
4	✗

b) Chandra Rakesh → Room Number {1 1000}



1 ≤ 1,000 → index

→ bool room[1001] = T

Check in room = 3:

room[3] = False

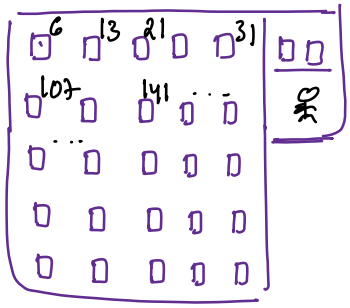
Check in room = 5:

room[5] = False

Check in room = 3: Occupied

TC: O(1)

c) Chandra Rakesh



{1000 lucky room number in range [1 - 10^9]

10^9, index → 10^9

bool r[10^9+1]

↳ Issues: Huge Space Wastage

↳ Advantage: TC: O(1)

Hashmap < Key, Value >

<79, occupied>
<85, occupied>
<93, occupied>
<113, occupied>

⇒ Check in (85)? Not available TC: O(1)?

TC: O(1) to search
SC: O(N) to store N entries

Adv content?

Note: keys are Unique

Value can be anything

Q1 Store population of every country:

Key: Country Name → String
Value: population → int / long

Key Value
HashMap < String, Long > hm

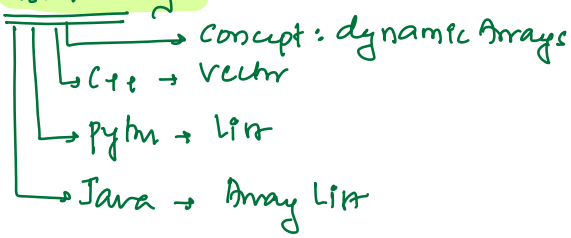
Q2 No: of States in Each Country

Key → Country Name → String
Value: → # No: of States → int

Key value
HashMap < String, int > hm

Q3 For every country we want to know all state names

Key: Country Name → Strings
Value: All State Names → List < Strings >



Key Value
HashMap < String, List < Strings >> hm;

Q4) For every country store population of each state

Key: Country Name → String

Value: population of each state → Hashmap<String, Long>

String → State name

Long → population

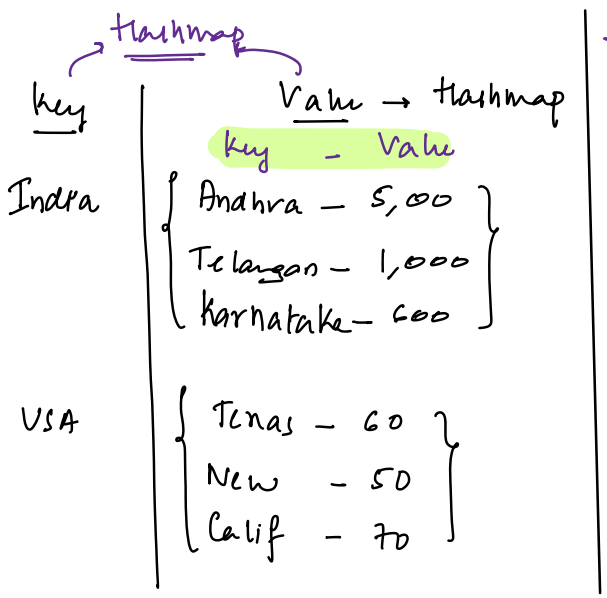
Hashmap<String, Hashmap<String, long>> hm

Key — Value

obs1: Value can be anything

obs2: Key can be only primitive datatype

{ int / long / float / double / String / char }



HashSet<Key>

→ We only store keys

→ keys have to be unique

Hashmap functnality

<Key, Value>

size : { Number of keys in hashmap }

insert (key, value)

search (key)

delete (key)

update (key, value)

A single operation will take O(1)

HashSet functnality

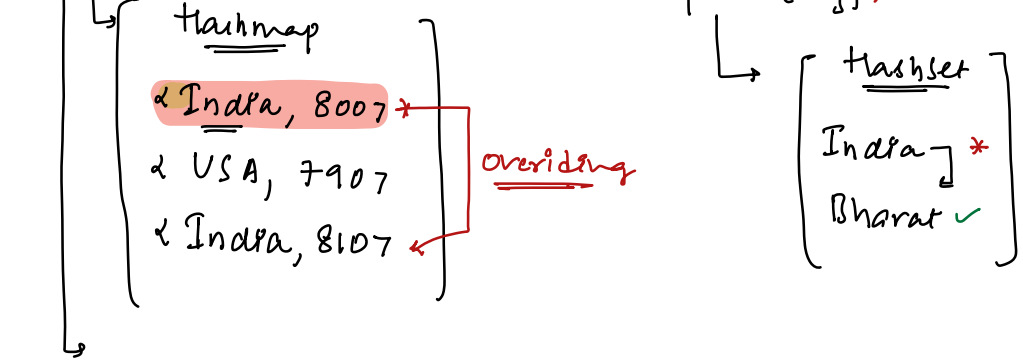
size : { Number of keys in hashset }

insert (key)

search (key)

delete (key)

update (key) *



Again insert <India, 8107>

[when we insert same key again it will override previous value]

Note:

- A single operation in hashmap / hashset ⇒ O(1)
- If we insert N <key, value>

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

→ { # Hashing library name in different languages }

Pseudocode	Java	C++	Python	JS	C#
Hashmap	Hashmap	unordered_map	Dictnmy	map	Dictnay
HashSet	HashSet	unordede-set	Set	Set	Hashset

10) Find Frequency of Numbers

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

Constraints:

$1 \le N \le 10^5$ $1 \le Q \le 10^5$ $1 \le arr[i] \le 10^9$

arr[10] = {2 6 3 8 2 8 2 3 8 10 6}

Q: 4

	freq
2 :	3 ✓
8 :	3 ✓
3 :	2 ✓
5 :	0 ✓

Idea1: For every query iterate & get count

→ TC: Q*N SC: O(1)

Idea2: Store data in Hashmap

Key → array elements → int

Value → freq of element → int

Hashmap<int, int> hm

hm<int, int>

<2, 3> <6, 2> <3, 2>

<8, 3> <10, 1>

```
void PrintFreq(int ar[], int Q[]){
    int n = ar.length // n → no: of elements
    int m = Q.length // m → no: of queries
    Hashmap<int, int> hm // TODO
    for(int i=0; i<n; i++){                    → O(N)
        if(hm.search(ar[i]) == True){          search
            // ar[i] is already present in hm
            hm[ar[i]] += 1 // update value by 1
        }
        else{
            hm.insert({ar[i], 1}) // insert
        }
    }
    for(int i=0; i<m; i++){                    → O(M)
        if(hm.search(Q[i]) == True){
            // get value of key = Q[i]
            print(hm[Q[i]])
                    → access, value of key = Q[i]
        }
        else{
            print(0)
        }
    }
}
```

Total TC:
O(N+Q)
SC: O(N)

28) Find the first non-repeating element, first element from start, non-repeating

ar[6] = { 1 2 3 1 2 5 }, ans = 3

ar[8] = { 4 3 3 2 5 6 4 5 } ans = 2

ar[7] = { 2 6 8 4 7 2 9 } ans = 6

Ideas:

1) Insert all elements in hashmap &
Iterate in iterate in hashmap to get 1st key with value = 1

ar[6] = { 1 2 3 1 2 5 }

Hashmap

<1,2>	<5,1>	<2,2>	<3,1>

Note: Order of insertion of keys is not maintained

2) Insert all elements in hashmap &
Iterate in array & get the first element with freq = 1

ar[6] = { 1 2 3 1 2 5 }

Hashmap

<1,2>	<5,1>	<2,2>	<3,1>

9:00 - 9:05am

Step 1: Insert all elements in hashmap → O(N)
Step 2: Iterate in arr[] get 1st element = 1 O(N)
with freq with 1

TC: O(N) SC: O(N) → TODO

4Q) Given ar[N] elements, find no of distinct elements

ar[5] = { 3 5 6 5 4 } ans = 4, each element consider 1 time

ar[5] = { 1 1 1 2 2 } ans = 2

ar[3] = { 3 3 3 } ans = 1

ar[7] = { 6 3 7 3 8 6 9 } ans = 5

Ideas: Insert all elements in hashset

ar[7] = { 6 3 7 3 8 6 9 }

hashset<int> hs → hs.size() = 5

6, 3, 7, 8, 9

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

Pseudo code:

```
hashset<int> hs
for(i=0; i<N; i++){
    hs.insert(ar[i])
}
return hs.size()
```

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

Q) Given ar[N] elements, check if all elements are distinct or not ?

ar[5] = { 6 8 3 2 7 } = {return True}

ar[7] = { 3 1 6 1 4 9 6 } = {return false}

Idea: → insert all elements in hashset

```
if (hashset size == N) {        TC → O(N)  SC → O(N)
    // all elements distinct
    return True
}
else {
    return false
}
```

Q) Given ar[N] elements, check if there exists a subarray with sum = 0

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

Idea

1) For every subarray calculate sum == 0, TC: O(N²) SC: O(1)

- O(N³) — 3 nested loops
- O(N²) — prefix sum
- O(N²) — carry forward

Idea 2:

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

Obs1: In pf[] numbers are repeating ⇒ There exists a subarray with sum = 0

pf[2] = 5 = sum[0 2]

pf[8] = 5 = sum[0 8] = sum[0 2] + sum[3 8]

5 = 5 + sum[3 8]

sum[3 8] = 0

pf[0] = 2 = sum[0 0]

pf[3] = 2 = sum[0 3] = sum[0 0] + sum[1 3]

2 = 2 + sum[1 3]

sum[1 3] = 0

Doubt:

ar[4] = { 2 -5 3 6 }

pf[4] = { 2 -3 0 6 }

↳ In pf[] no repeatim, but subarray with sum = 0

pf[2] = sum[0 2] = 0

↳ Subarray

Note: In your pf[], even if single zero present, then exists a subarray with sum = 0

Final Obs: → If element repeat in pf[] → There exists a subarray with sum = 0

→ If 0 is present in pf[]

```
bool subarrayZero(int ar[]){ TC→O(N) SC→O(N)
    int n = ar.length
    int pf[n] // Construct pf[] TODO
    HashSet<Int> hs;
    for(int i=0; i<N; i++){
        if( pf[i]==0) {return True}
        hs.insert(pf[i])
    }
    if( hs.size()<N) {// repeation in pf[]
        return True
    else {return false}
}
```

// run as custom input

↳ print( )

// Constant / Test

↳ ar.length;