# Hashing 1 : Introduction

Radisson Hotel

5 rooms only

Register

1 Yes
2 No
3 :
4 :
5 :

5 room → 1000 rooms [1, 1000]
labelled as
↳ occupied / not occupied

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

Pandemic
↓
$[1 - 10^9]$

bool room[$10^9 + 1$] → Issues: Huge space wastage
→ Advantage: TC: O(1)

Hashmap stores <key, value> pair

<10015, occupied>
<123, occupied>
<129, not occupied>
⋮

⇒ check in 10015?
⇒ occupied in O(1) tc

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

Hashmap<int, bool>

Note : Keys are unique
value can be anything

Question :
Store population of every country

key : country name → string
value : population → int / long

Hashmap <string, long> hm

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

<India, UP>

<India, MP> ✗ Keys have to be unique

HashMap<String, list<String>> hm

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 be primitive datatype

Hashmap functionality

size: {# keys present}

insert (key, value)

search (key)

delete (key)

update (key, value)

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 the freq. of given element in the array.

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

Q = 4 freq

2 : 3

8 : 3

3 : 2

5 : 0

constraints

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

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

Idea 1 : for every query, iterate & get count

TC : O(Q x N)  SC : O(1)  not feasible

Idea 2 : Store freq in hashmap

Key → array element : int

value → freq. of element : int

HashMap < int, int >

{ 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> um

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

  if(um.search(a[i]) == true){
      // a[i] is present
      um[a[i]]++  // update +1
```

```
        }
        else {
            hm.insert({a[i], 1}) // insert
        }
    }
}
for(i=0; i<B; ++i){
    if(hm.search(input[i]) == true){
        print(hm[input[i]])
    }
    else {
        print(0)
    }
}
```

TC: O(N+B)

SC: O(N)

Question2: find the first non repeating element.

a[6] = { 1 2 3 1 2 5 } ans = 3

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

Idea 1

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

order of insertion of keys is not maintained in hashmap/ hashset

Idea2:

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

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

## Question 3

Given N elements, find no. of distinct elements.

a[5] = { 3 5 6 5 4 }  ans = 4

Insert everything in hashset & print its size.

hashset will not insert duplicate entries

Code

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

TC : O(N)

SC : O(N)

## Question 4

Given N elements, check if there exists a subarray with sum 20

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

ans = true

| | 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 |

Observation) : In pf[], numbers are repeating ?

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

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

pf[] = {2 -3 0 6}

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

$\because pf[2] = 0 \Rightarrow Sum[0-2] = 0$

Note: In your pf() even if single 0 is present, there exist a subarray with sum=0

Final Idea

If ele. repeat in pf()
OR
If 0 present in pf()
→ there exist subarray with sum=0

Code

```
bool zeroSum (int a[], n) {
    pf[n]  // create pf array - TODO
    HashSet<int> hs
    for(i=0 to n-1) {
        if (pf[i] == 0) { return true }
        hs.insert(pf[i])
    }
```

```
if (us.size < n) {        // repetition in ptr
    return true
}
return false
}
```

TC: O(N)

SC: O(N)