Content →

- Hashing
- Distinct elements
- Frequency of an element
- First non repeating
- check subarray with sum=0

Friday
Contest
Math, Recursion
OOPS

Hashing

Hotel   10 rooms   Register

Software

1000 rooms   bool ar[1000]

10^9 rooms   bool ar[10^9]

⇒ Only some are occupied

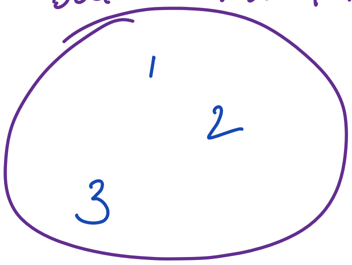
10^5 guests ⇒ only 10^5

wasting majority of space of array

How to not waste space?

⇒ Only maintain the occupied room no.s

Data structure



insert
deletion   O(1)
finding

Hashset

unique
values

Key - only

Hashmap

<10, "India">
<20, " USA">
<30, " UK">

key - value pair

1) Store population of every country

country → population

String int/long

2) For every country, number of states

country → no of states

string int

3) For every country, store all state names

country → all state names

string → list<string>

HM can be used inside another HM

student → subject wise marks

string → hm < string, int >

country → hm< string , int>

Note: Value can be ANYTHING

Key: { primitive datatype

string, int, float, char, }

boolean.

→ 10

Hashmap < key, value >
insert (key, value)
search (key)
remove (key)
size
getValue (key)

All operations are O(1)

Hashing (Purely mathematical)

Can key have multiple values NO

Hashset < key >
insert (key)
remove (key)
search (key)
size

All O(1)

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

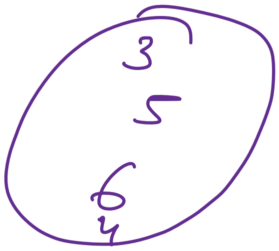
Q1) Given N array elem, find no of distinct elements

Eg A[6] = [3, 5, 6, 5, 4, 3]   ans = 4

Idea : Here use hashset

Obs : Insert everything into hs
ans = hs.size()

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



TC } O(N)
SC }

Q2 Find frequeny of numbers

Given N numbers & Q queries, for each query find frequency of that number

Eg: A[10] = [2, 6, 3, 8, 2, 8, 2, 3, 8, 10]

Q[4] = [2, 8, 3, 5]

number → freq

Hashmap<int, int> hm

2 : 3
8 : 3
3 : 2
5 : 0

HM

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

Code

```
List<int> frequency ( int arr[], int Q[] ) {
    hashmap<int, int> hm
    for (i=0 ; i<n ; i++ ) {
        if ( hm.contains (ar[i]) == true)
            hm [ar[i]] ++                //value +1
        else
            hm.insert (ar[i], 1)
    }
    list<int> ans
    for (i=0 ; i<q ; i++) {
        if (hm.contains ( Q[i]) == true)
            ans.add ( hm[Q[i]] )
        else
            ans.add ( 0)
    }
    return ans
```

Q3 Find first non repeating elem

Eg 1 A[6] = [1, 2, 3, 1, 2, 5] ans = 3

A[8] = [4, 3, 3, 2, 5, 6, 4, 5] ans = 2

Idea: 1) Create frequency hashmap
2) Iterate through array.
if freq > 1 continue
if freq = 1 this is my ans

Code

```
int nonrepeat ( int arr[] ) {
    1) create frequency hashmap
       hashmap<int, int> hm
       for (i=0 ; i<n ; i++ ) {
           if ( hm.contains (arr(i)) == true )
                  hm [arr[i]] ++          // value +1
           else
                  hm. insert ( arr[i], 1 )
       }
 2) Iterate through array
       for ( i=0 ; i<n ; i++ ) {
           if ( hm [ arr [i] ] == 1) {
                  return arr [i]
           }
       }
 return -1
```

SC }
TC } O(N)

Q4 Check if there exist subarray with sum = 0

Eg1 A[7] = [2, 3, -1, 4, -3, 10, 4]

Eg2 A[5] = [1, 2, -1, -2, 4]

Idea: PF sum of subarray [s:e] = pf[e] - pf[s-1]

we want this to be 0.

$\Longrightarrow$ pf[e] = pf[s-1]

Obs: 1) any prefix sum entry repeats
2) any prefix sum entry = 0

Code

```
bool subzero ( int arr[] ) {
    1) Create pf array

    2) Create freq hashmap on the pf array

    3) if ( hm.contains (0) == true )
            return true

        for ( i=0 ; i< n ; i++) {
            if ( hm [ pf[i] ] > 1)
                return true
    }
    return false
}
```

A[7] = [2, 3, -1, 4, -3, 10, 4]
pf 2 5 4 8 5 15 19

A[5] = [1, 2, -1, -2, 4]
pf 1 3 2 0 4

2	1
5	3
4	1
8	1
15	1
19	1

A[8] = [2, 3, -1, 4, -3, 10, 4, -14]
pf 2 5 4 8 5 15 19 5

{dou}

pf[i] , i

A[7] = [2, 3, -1, 4, -3, 10, 4]
pf 2 5 4 8 $\underline{5}$ 15 19

e = 4
s - 1 = 1
s = 2

2 → 0
5 → $\underline{1}$
4 → 2
8 → 3