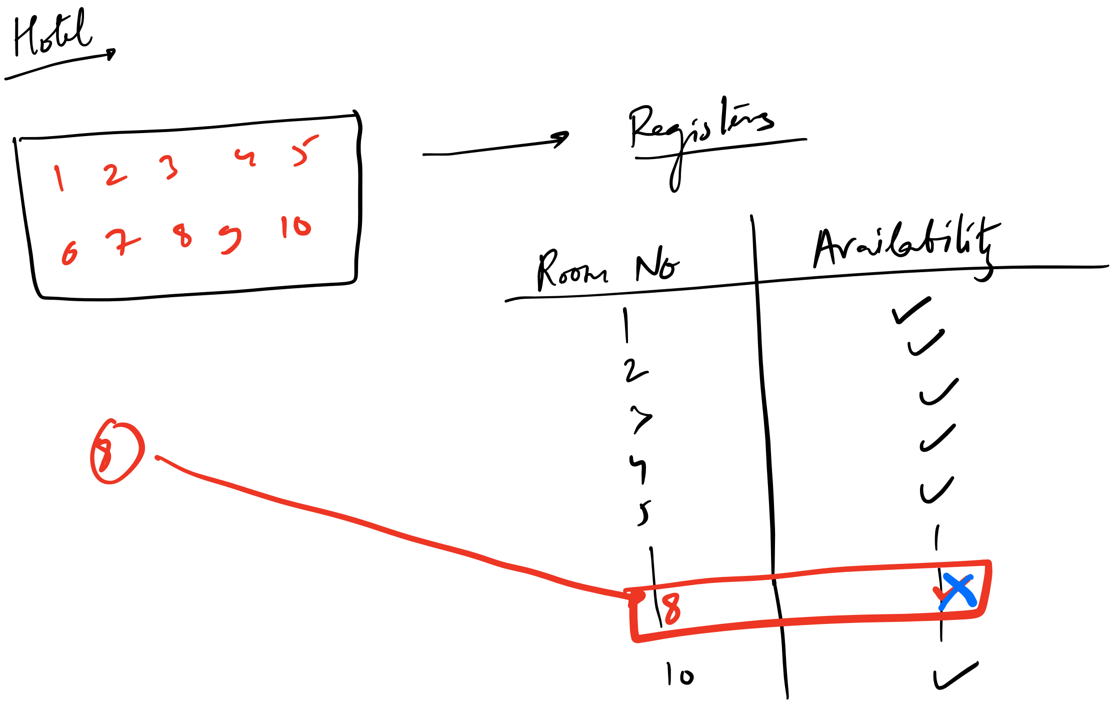
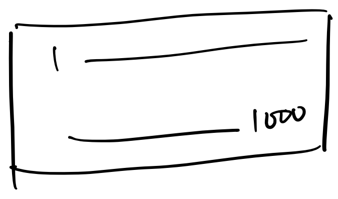
@ Hotel



(b)



Room No: [1, 1000]

bool Av [1001] → { True }

$\downarrow$

0 — 1000

```
x → if ( Av[x] == true) {
        → room is Avail--
        Av[x] = false;
    }
    else {
        → room is not Avail--
    }
```

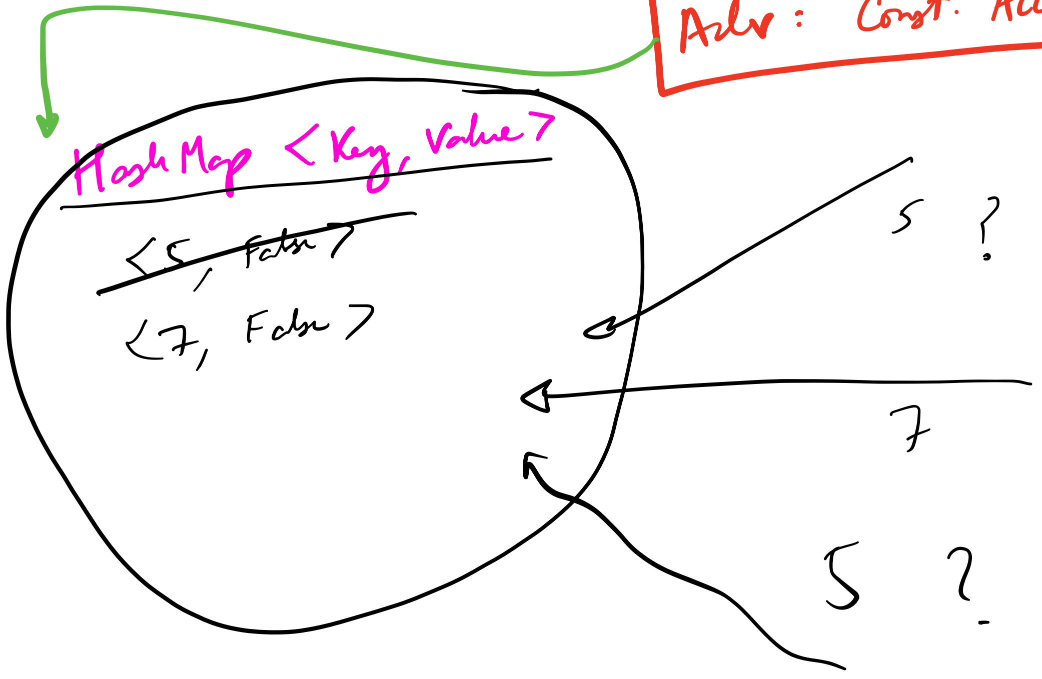
(c) Lucky no's : [7, 12, 37, 45, ——] 1000 lucky no's

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

7, 12, 37, 45
—— ——
————

bool Ar[10^9+1]; ✗

Issue : Wastage of space
Adv : Const. Access time



Check/Update : O(1)

$\boxed{SC : O(N)}$: N → No. of rooms

$\boxed{\text{Keys : Unique}}$

Q Store population of every country!

$\underset{\text{Key}}{\text{HM < Country Name}}$, $\underset{\text{Value}}{\text{population}}$ >

HashMap < string, long > hm;

Q No. of states for every country!

$< \underbrace{\text{country Name}}, \underline{\text{No. of states}} >$

↓

HashMap < string, int > hm;

Q $\underbrace{\text{for every country, store all state names}}$!

< Country Name, list of state Names >

HashMap < string, List < string >> hm;

Q. for every country, store the pop. of every state!

HashMap<Country Name, HM<stateName, pop>> hm;

Key	Val
INDIA	→ {{KA → 990}, {MR → 1}, {MN → 5}}
US	→ {{AR → 100}, {TX → 50}}

HashMaps < Key, Value >

Key → int, float, double, long, char, strings,

Value → ANYTHING!

HashMap functionality

< K, V >

insert < K, V >
search / find / get < K >
contains < K >
delete < K >
update < K, V' >
size → # Keys

} O(1)

SC for N keys : O(N)

Hash Set

< K > → Unique

insert < K >
delete < K >
contains < K >
size

} O(1)

SC for N keys : O(N)

HM / HS name in diff. languages

	JAVA	C++	Py	C#	JS
HM	HashMap	unordered_map	dict	dict	map
HS	HashSet	unordered_set	set	hashset	set

NOTE:

HashMap & HashSet
do not have Keys sorted !

~~Q~~ Given an Array & Q queries.
For every query, find the frequency

A : | 1 5 1 2 3 2 1 1 1 |

$1 <= A[i] <= 10^9$
$1 <= N <= 10^5$

$2 \longrightarrow 2$

$7 \longrightarrow 0$

$1 \longrightarrow 5$

< Elem, freq > → < int, int >

{ < 2, 2 >
< 5, 1 >
< 3, 1 >
< 1, 5 > }

```
HashMap <int, int> hm;

f ( i = 0; i < N; i++) {                     ——→ N   : O(N)
    if ( hm.contains(A[i]) == true) {        → O(1)
        old freq = hm.get(A[i]);
        new freq = old freq + 1;             → hm[A[i]]++;
        hm.insert( A[i], new Freq);
    }
    else {
        hm.insert( A[i], 1);
    }
}
```

```
f( i=1; i<=Q ; i++) {          ——→ Q     : O(Q)
    //x
    if( hm.containsKey(x) == true) {     ——→ O(1)
        print( hm.get(x));               ——→ print(hm[x]);
    }
    else {
        print(0);
    }
}
```

$\boxed{TC : O(N+Q)}$

$\boxed{SC : O(N)} \longrightarrow O(\text{Distinct elements in } A)$

Q Given an array. Find the first (from left) non-repeating element!

A : [1, 2, 3, 1, 2, 5]

Idea: 1. Create a freq HM

2. Iterate on Array L→R
find the 1st element with
<u>freq == 1</u>

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

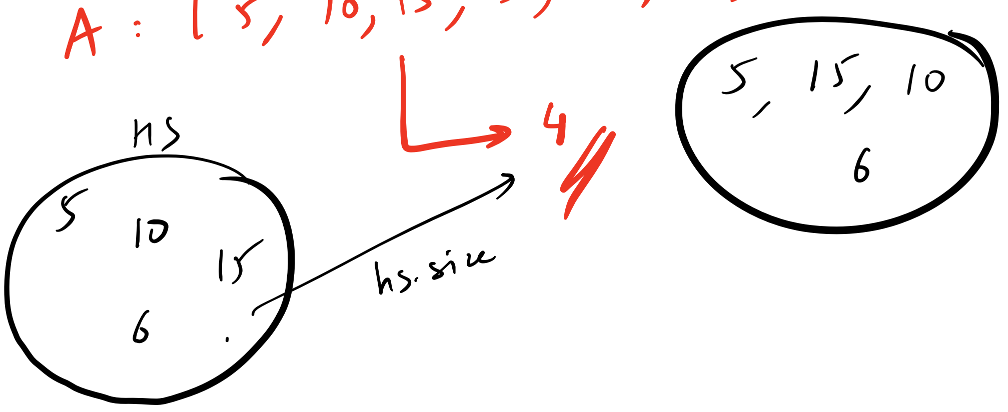
```
HashMap < int, int > hm;
f (i: 0 ⟶ N-1) {            ⟶ N        | O(N)
    hm[A[i]]++;             ⟶ O(1)
}
f (i: 0 ⟶ N-1) {            ⟶ N        | O(N)
    if ( hm[A[i]] == 1 ){   } ⟶ O(1)
        ret A[i];
    }
}
ret -1;
```

$\boxed{TC: O(N)}$

$\boxed{SC: O(N)}$

Q Given array. Find the no. of DISTINCT elements.

A : [5, 10, 15, 5, 10, 6]



```
HashSet <int> hs;
f (i=0 → N-1) {                → N        | O(N)
    hs.insert(A[i]);           → O(1)
}
ret hs.size();
```

$\boxed{TC = O(N)}$

$\boxed{SC = O(N)}$

Q Given an array.
Check if all values are different than each other. → DISTINCT !

[3, 1, 7, 4, 5] → true !

[3, 1, 7, 1, 4, 5] → false !

I Idea:
Insert all in HashSet
if(hs.size() == N) → ALL DISTINCT

else → NOT !

```
HashSet < int > hs;
f(i : 0 ⟶ N-1) {          ⟶ N     | O(N)
        hs.insert( A[i]);   ⟶ O(1)

}
if( hs.size() == N) ret true;
ret false;
```

$\boxed{TC = O(N)}$

$\boxed{SC : O(N)}$

II

Idea: <u>fail fast</u>!

```
HashSet <int> hs;
  f(i:0 ⟶ N-1) {
     if( hs.contains( A[i]) == true) {
              ret false;

     }
     hs.insert( A[i]);
  }
  ret true;
```

$$\boxed{\begin{matrix} TC \\ SC \end{matrix} \rightarrow O(N)}$$

Q Given an Array. Check if there is **ANY subarray with SUM == 0**

$$A: \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ [2, & 2, & 1, & -3, & 4, & 3, & 1, & -2, & -3, & 2] \end{array} \rightarrow \text{True}$$

I) BF

$\forall$ Sub Array $\longrightarrow N^2$

find Sum & check $\} \longrightarrow N$

$\boxed{TC : O(N^3)}$
$\boxed{SC : O(1)}$

II P.S

Build PS $\longrightarrow N$

+

$\forall$ S.A s $\longrightarrow N^2$

find Sum & check $\} \longrightarrow O(1)$

$\boxed{TC : O(N^2)}$
$\boxed{SC : O(N)}$

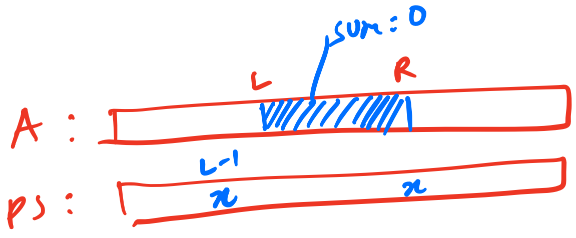
III Carry forward !

A : [2, 2, 1, -3, 4, 3, 1, -2, -3, 2]

~~$\boxed{TC : O(N^2)}$~~

~~$\boxed{SC : O(1)}$~~

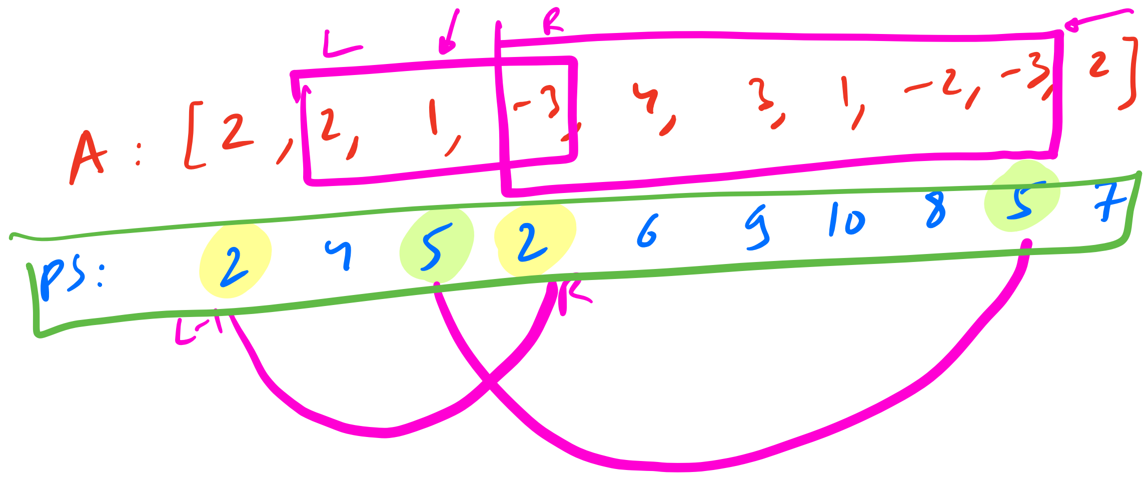
IV

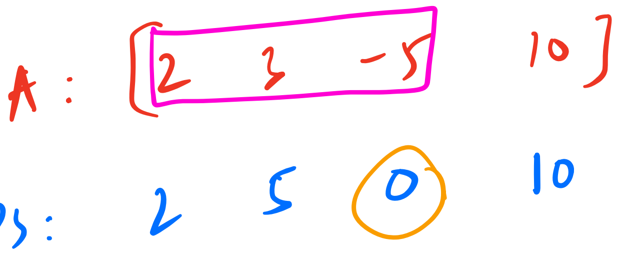


$$\text{sum}(L, R) = ps[R] - ps[L-1]$$

$$\underbrace{\qquad}_{0} = ps[R] - ps[L-1]$$

$$\boxed{ps[L-1] = ps[R]}$$





```
//A[], N

HashSet<int> hs;
hs.insert(0);

ps=0;
f(i:0 → N-1){              ———— N
    ps += A[i];
    if( hs.contains( ps) == true) {
            ret true;

    }
    hs.insert( ps);
}
ret false;
```

$\boxed{TC = O(N)}$

$\boxed{SC = O(N)}$