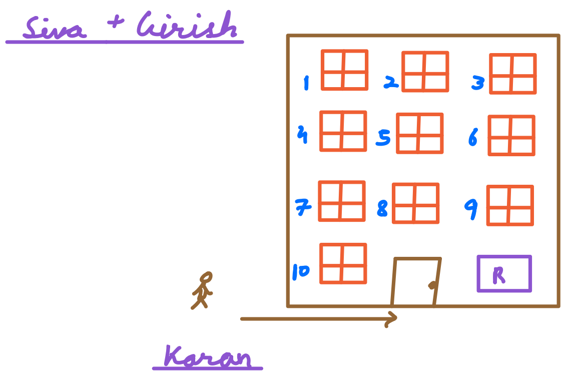

Room No.	Free
1	✗
2	✓
3	✓
⋮	⋮
10	✗

→ optimized

1) Hardwork of going to every room.
2) Waiting time of guest.

HashTable

< Key , Value >

unique — Room No. — Free — useful information about the key.

Room No. → AC, Balcony, TV, Bath tub, wifi, Geyeser, Sofa, etc. → value

Key

	Key	Value
1) opulation of every country →	Country Name String	Population Long
2) Count of cities of every country →	Country Name String	Count of cities int
3) All city names of every country —	Country Name String	List of city names List<String>
4) Population of every city for every country →	Country Name → String	<City , Population> String long

---

HashMap

< key , value >

unique

1) insert < key, value > . → hm.put(K, V)
2) update value for a given key . → hm.put (K, V)
3) delete < key, value > for a given key. → hm.delete (K)
4) search for a value given the key. → V = hm.get(K)
5) Size of hashmap. → hm. size ()

Internal Implementation idea
→ Advance content

All functions
→ TC = O(1)

6) Key is present or not. → hm.containsKey(K)

7) Map is empty or not. → hm.isEmpty()

HashSet <key> → unique

Insert

< Vinit, 100 >
< Tejas, 47 >
< Deepanshu, 100 >
< Girish, 67 >
< Abhishek, 91 >
< Vikram, 100 >

Travel & Print → It will print the data in random order.

Q→ Given an integer array, find frequency of an element of the array for multiple queries. Query → x (int)

A = [2 6 3 8 2 8 2 3 3]

- =	0	0	3	3	0	0	1	0	2
	0	1	2	3	4	5	6	7	8

F → i

Queries

x	freq(x)	
8 →	2	✓
3 →	3	✓
10 →	0	←

```
for i → 0 to (N-1)
        F[A[i]] += 1
```

TC = O(N)

SC = O(Range of A[i])

A[i] ≈ 10^9 → MLE error

< Key , Value >

A[i] , freq(A[i])

2 → ~~1~~ ~~2~~ 3

6 → 1

3 → ~~1~~ ~~2~~ 3

8 → ~~1~~ 2

SC = O(N)

A = [2 6 3 8 2 8 2 3 3]

→ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

```
for i → 0 to (N-1)
    if (hm.containsKey(A[i]))
        f = hm.get(A[i])
        hm.put(A[i], f+1)
    else
        hm.put(A[i], 1)
```

TC = O(N)

```
for i → 1 to Q
    // Read x
    if (hm.containsKey(x))
        print (hm.get(x))
    else
        print(0)
```

$TC = O(Q)$

Total $TC = \underline{O(N+Q)}$

Q → Count no. of distinct elements present in the array.

A = [6 3 7 3 8 6 9]   Ans = $\underline{5}$

A = [8 8 8 8 8]   Ans = $\underline{1}$

```
for i → 0 to (N-1)
    hs.add(A[i])
return hs.size()
```

Set < A[i] >

Map

< Key , Value >

A[i]   anything ✓

count/freq. ←

isPresent ←

$TC = O(N)$

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

{6, 3, 7, 8, 9} → size = $\underline{5}$

10:45 PM

Q → Find first non-repeating element in A[], if not present return -1.

✗ ✗ ✓

A = [1 2 3 2 5 1 2 8]   Ans = $\underline{3}$

0 1 2 3 4 5 6 7

✓

A = [1 2 3 2 3 6]   Ans = $\underline{1}$

0 1 2 3 4 5

A = [1 2 4 1 2 3 4 3]   Ans = $\underline{-1}$

non-repeating element → freq = 1

Map < A[i], freq of A[i] >

1 → 2

2 → 3

3 → 1

A = [1 2 $\boxed{3}$ 2 5 1 2 8]

0 1 2 3 4 5 6 7

```
for i → 0 to (N-1)
    if (hm.containsKey(A[i]))
        f = hm.get(A[i])
        hm.put(A[i], f+1)
    else
        hm.put(A[i], 1)
```

5 → 1
8 → 1

$TC = O(N+N) = \underline{O(N)}$
$SC = \underline{O(N)}$

```
for i → 0 to N-1
    if (hm.get(A[i]) == 1)
        return A[i]
return -1
```

Q → Given an integer array A,
check if there exists a subarray with sum = 0.
continuous part of array

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

Ans = true

Bruteforce → ∀ subarrays, check if sum = 0.
$TC = \underline{O(N^2)}$ $SC = \underline{O(1)}$

Observation → Subarray Sum → use Prefix Sum.

			i						j	
	0	1	2	3	4	5	6	7	8	9
A =	[2	2	1	-3	4	3	1	-2	-3	2]
P =	[2	4	5	2	6	9	10	8	5	7]

∀i P[i] = P[i-1] + A[i]

A[0]+A[1]+A[2] = 5

A[0]+A[1]+A[2] +
0 ← A[3]+A[4]+A[5]+A[6]+A[7]+A[8] = 5
index (2+1) —— 8

x = 5
x + y = 5 } ⇒ y = 0

P[i] == P[j] ⇒ subarray
(i+1)—j sum = 0

↓
Ans = true

Solution Steps → 1) calculate prefix sum

2) count # distinct elements (P[ ]) < N ⇒ Ans = true

A = [3 -1 2 -4 -2]

P = [3 2 4 0 -2]

↳ ∵ prefix is also a subarray

if (P[i] == 0) ⇒ Ans = true.

A = [0 8 3]

P = [0 8 11]

```
P[0] = A[0]
if (P[0] == 0)
    return true
for i → 1 to (N-1)
    P[i] = P[i-1] + A[i]
    if (P[i] == 0)
        return true

for i → 0 to (N-1)
    hs.add(P[i])
if (hs.size() < N)
    return true

return false
```

TC = O(N)

SC = O(N)

A = [5 6 0 2 3]

P = [5 11 11 13 16]

distinct elements = 4 < N

⇒ Ans = true