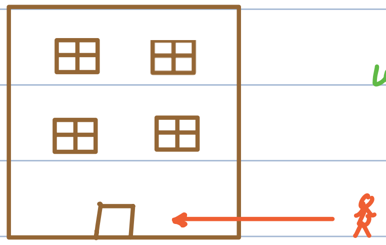
Areur



Room 5

Key → Value → can be set of entries

unique

Room	Availab., AC, Capacity ...
1	✓
2	×
3	×
⋮	

rooms increases to 10K

Will data in register work ?

No → We need to use tech.

HashMap stores data → < Key , Value >

(internal implementation → later)

Store population of every country

Hashmap < String , Long >

< Country , Population >

Store # states of every country

Hashmap < String , Int >

Country , #states

Store the name of all states of every country.

Hashmap < String , List < String >>

Country list of all states

Population of each state of every country

Hashmap < String , Hashmap < String , Larg >>

country   state, population

Hashset < Key > → Used to store unique data.

Functionalities

HashMap

1) insert (key, value) → put (K, V)
2) size ()
3) delete (key)
4) update (key, value) → similar to second insertion ∵ for same key value will be overriedden.
5) search (key)

} TC = O(1)

Hashset

1) insert (key)
2) size ()
3) delete (key)
4) search (key)

} TC = O(1)

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

H.W → check syntax in your language.

Hashmap/Hashset → Order of insertion is not maintained.

Treeset / Treemap → Order is sorted wrt keys.

Linked Hashmap/ Linked Hashset } → Order of insertion is maintained.

insert → 1 5 8 stored → 1 5 8

Q → Given an integer array & Q queries where we have to find frequency of the given element.

$$A = \left[\begin{array}{ccccccc} 0&1&2&3&4&5&6\\ 2&6&3&8&2&8&2 \\ \checkmark & \uparrow & & & \checkmark & & \checkmark \end{array}\right]$$

Query →

$2 \rightarrow 3$

$6 \rightarrow 1$

$5 \rightarrow 0$

Bruteforce → ∀query, travel & check the frequency. TC = $O(Q * N)$

Frequency array F[i] = freq. of i

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

SC = $O(|A[i]|)$ $1 <= A[i] <= 10^9$

(MLE)

Sol → Use hashmap to store data.

```
    // hm <int, int>
for i → 0 to (N-1) {
    if (hm.containsKey(A[i]))
        hm.put(A[i], hm.get(A[i]) + 1)
    else
        hm.put(A[i], 1)
}
for i → 0 to (Q-1) {
    if ( hm.containsKey (A[i]))
        print ( hm.get (A[i]))
    else
        print (0)
}
```

TC = O(N+Q)   SC = O(N)

Q → List of learner id wrt each attempt.
Find learners with least attempts.

id → [101 102 103 101 102 101 105]

least participation → 1
id → {103, 105}

1) Find freq ∀ ids
2) Find least value
3) Keys with least values give list of learners.

Q → Find the first non-repeating element.
only present once

N = 6   A = [1 2 3 2 5 1]
(Ans)

A = [4 3 3 2 5 6 4 5]
(Ans)

idea → use & store frequency

1 ⇒ non-repeating

Sol → 1) Store freq ∀ elements

```
   // hm <int, int>
  for i → 0 to (N-1) {
        if ( hm.containsKey (A[i]))
              hm.put (A[i], hm.get (A[i]) + 1)
      else
              hm.put (A[i], 1)
  }
```

2)

```
  for  i → 0 to (N-1) {  // index traversal ✓
        if ( hm.get (A[i]) == 1)
              return A[i]
  }
  return -1
```

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

Q → Given an integer array, find the count of distinct elements.

$$A = \left[\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 5 & 6 & 5 & 4 & 6 & 5 \end{array}\right]$$

{3 5 6 4}   Ans = $\underline{4}$

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

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

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

Q → Given an integer array, check if there is a subarray with sum 0.

$$A = [\;\overset{0}{2}\;\; \boxed{\overset{1}{2}\;\; \overset{2}{1}\;\; \overset{3}{-3}}\;\; \overset{4}{4}\;\; \overset{5}{3}\;\; \overset{6}{1}\;\; \overset{7}{-2}\;\; \overset{8}{-3}\;\; \overset{9}{2}\;]$$

Ans = $\underline{\text{true}}$

<u>Bruteforce</u> → ∀ subarrays, calculate sum & check.

TC = $O(N^3) \to \underline{O(N^2)}$ SC = $\underline{O(1)}$

sum of subarray → prefix sum

$i - j$ $P[j] - P[i-1] = 0$

$\Rightarrow P[j] = P[i-1]$

case 1 → $P[i] = 0 \Rightarrow$ Ans = true ✓

2 → $P[j] = P[i-1]$ repeating value in P[] ✓

$$A = [\;\overset{0}{2}\;\; \boxed{\overset{\overset{i}{1}}{2}\;\; \overset{2}{1}\;\; \overset{\overset{j}{3}}{-3}}\;\; \overset{4}{4}\;\; \overset{5}{3}\;]$$

$$P = \underset{0}{\textcircled{2}}\;\; \underbrace{4\;\; 5\;\; \textcircled{2}}_{3}\;\; 6\;\; 9$$

```
P[0] = A[0]
if (P[0] == 0) return true ✓
hs.add(P[0])
for i → 1 to (N-1) {
```

$A = [\underline{2\;\; 3\;\; -5}\;\; \ldots]$

$P \to 2\;\; 5\;\; 0$

```
        P[i] = P[i-1] + A[i]
        if (P[i] == 0) return true
        if (hs.contains(P[i])) return true
        hs.add(P[i])
    }
    return false
```

TC = O(N)   SC = O(N)

Q → Find the count of subarrays with sum 0.

```
cnt = 0
P[0] = A[0]
if (P[0] == 0)  cnt++
hs.add(P[0])
for i → 1 to (N-1) {
    P[i] = P[i-1] + A[i]
    if (P[i] == 0)  cnt++
    if (hs.contains(P[i]))  cnt++
    hs.add(P[i])
}
return cnt
```

TC = O(N)   SC = O(N)

H.W → Find count of subarrays
with sum K.
