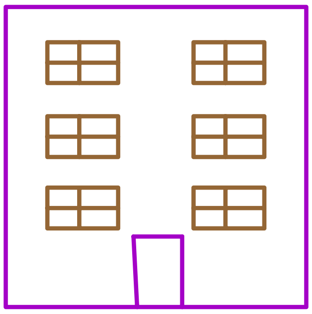
Avg PSP 74 → 77 → 80
✓

Amit



Hash Table/Hash Map

<Key, Value>

Room	Free
1	✓
2	×
3	✗
⋮	⋮

← Sonal (5 is available)

Same Room No. → confusion ⇒ problems & conflicts.

⇒ Key have to be unique

Value → useful information about the key.

Room → capacity, AC, wifi, balcony, etc.

⇒ one key can have multiple values

Q → 1) Store population of every country.

Key → Country Name (strings)
Value → Population (long)

2) Store # cities of every country.

Key → Country Name (strings)
Value → # cities (int)

3) Store list of cities of every country.

Key → Country Name (strings)
Value → list of cities (List <String>)

4) Store population of every city of every country.

Key → Country Name (strings)

Value → Population of every city (HashMap <String, int>)

Value → anything ✓

Hashset (value = null)

→ only unique keys.

Internal Working → Advanced DSA ✓

Functions

1) Insert/Put → <Key, Value>
(update)
2) Size
3) Remove → <Key>
4) Search/Get → <Key>

TC = O(1)

Java	C++	Python	C#	JS
HashMap	unordered_map	Dictionary	Dictionary	map
HashSet	unordered_set ...	set	HashSet	set

1) insert (a, 1)
2) insert (b, 2)
3) insert (c, 3)

→ travel the map & print keys

order will not be same as order of input.

o/p → c b a OR a c b OR a b c

Q → Find the frequency of element for multiple queries in the array.

$$A = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 6 & 3 & 8 & 2 & 6 & 8 \end{array}$$

Query	freq
6 →	2
3 →	1
5 →	0

Bruteforce → ∀ queries, iterate & count the frequency.

$TC = \underline{O(Q * N)}$ $SC = \underline{O(1)}$

Frequency Array

$\boxed{F[i] = \text{frequency of element } i}$

$$\rightarrow A = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 6 & 3 & 8 & 2 & 6 & 8 \\ \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark \end{array}$$

maxA = 8

$$F = \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 0 & 0 & \cancel{0} & \cancel{0} & 0 & 0 & \cancel{0} & 0 & \cancel{0} \\ & & \cancel{1} & 1 & & & \cancel{1} & & \cancel{1} \\ & & 2 & & & & 2 & & 2 \end{array}$$

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

for i → 0 to (Q-1) {
        x = Query[i]
        print (F[x])
}
```

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

If range is high ≈ 10^9

$F[10^9]$ → overflow

Hash Map < Key, Value >

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

// mp<int, int>

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

```
for i → 0 to (Q-1) {
    x = Query[i]
    if (mp.containsKey (x )) {
        print (mp.get(x))
    } else {
        print(0)
    }
}
```

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

10:35 PM

Q → Find the first non-repeating element in array.

unique (freq = 1)

Eg → $A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 1 & 5 & 2 \\ ✗ & ✗ & ✓ & & & \end{array}$ Ans = 3

Sol → 1) calculate freq ∀ A[i]. ✓

2) Travel the ~~map~~ i/p array & get first element with freq = 1 as answer.

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

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

Q → Count the # distinct elements in the array.

A = [3 5 6 5 6] → {3, 5, 6}   Ans = 3

A = [3 5 5 3 5] → {3, 5}   Ans = 2

value is not required ⇒ HashSet

```
// hs → HashSet
for i → 0 to (N-1) {
        hs.insert(A[i])
}

return hs.size()
```

TC = O(N)

SC = O(N)

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

$$A = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ [2 & 2 & 1 & -3 & 4 & 3 & 1 & -2 & -3 & 2] \end{array}$$

Ans = true

subarray sum → prefix sum

$$\boxed{P[j] = \sum_{i=0}^{j} A[i]}$$

subarray sum(i — j) = P[j] − P[i-1] = 0

⇒ P[j] = P[i-1]   Ans = true

sum(0 — j) → $\boxed{P[j] = 0}$

```
// hs → HashSet
for i → 0 to (N-1) {
  → if (P[i] == 0)   return true
```

```
    → if (hs.contains(P[i])) return true
    → hs.insert(P[i])
}
return false
```

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

	0	1	2	3	4	5
A = [	2	5	2	-4	2	-5]
P = [	2	7	9	5	7	

H/S: 2 7 9 5

H.W → count #subarrays with seem = 0.