Hotel Business

Raushan Prakhar

→ Maintain Register

Expanding 1000 rooms

Room No.	Occupied
1	F
2	F
3	F
4	F
5	F

① Room nos. <1-1000>

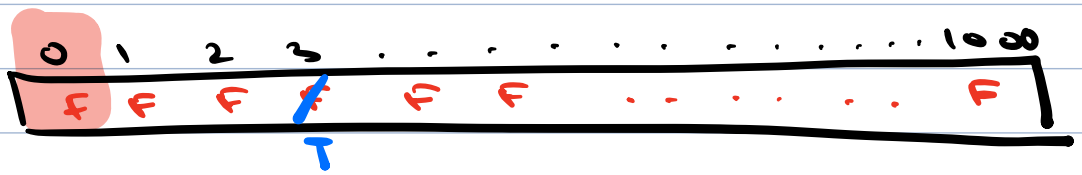
array → idx → room no.

→ value → occupancy status

bool isOccupied [1001]

Occupied → T

else → F



Status can be checked in O(1)

Update status → O(1)

② Room no. <1-n>

bool isOccupied [n+1]

idx
0 → n

③ Pandemic hit → business went down

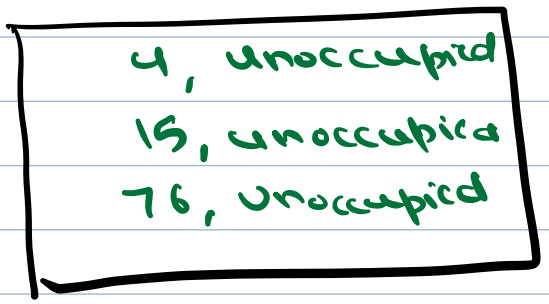
↓

1000 lucky nos. in range [1-10^9]

< 4, 15, 76, ...>

1000 ids are used of 10^9

// Hashmap is a data structure that store <key, value> pairs



Room, occupancy status
↓ key ↓ value

In Hashmap, Tc of search is O(1) and Sc is O(n)
↳ no. of entries

Hashmap <key, value>
↓ unique ↓ can be anything

// If we want to store only keys →
Hashset <key>
↳ key → unique
search ↓ O(1) in Hashset

// value - can be anything
// key - <String, int, long, float, . . . >
↓ Any primitive data type

↳ a list can't be kept as a key

1) Store population of every country

unique

Hashmap < key , value >

key → Country.

value → Population

int → 10^9

long → 10^{18}

Hashmap < String , long >

2) No. of states in each country

Hashmap < key , value >

key → Country

value → no. of states

Hashmap < String , Int >

3) For every country, we want to store all state names

Hashmap < key , value >

key → Country

value → List of state names

Hashmap <String , List <String >>

hm [key] → return value

4) For every country, store population of each state.

Hashmap < key , value >

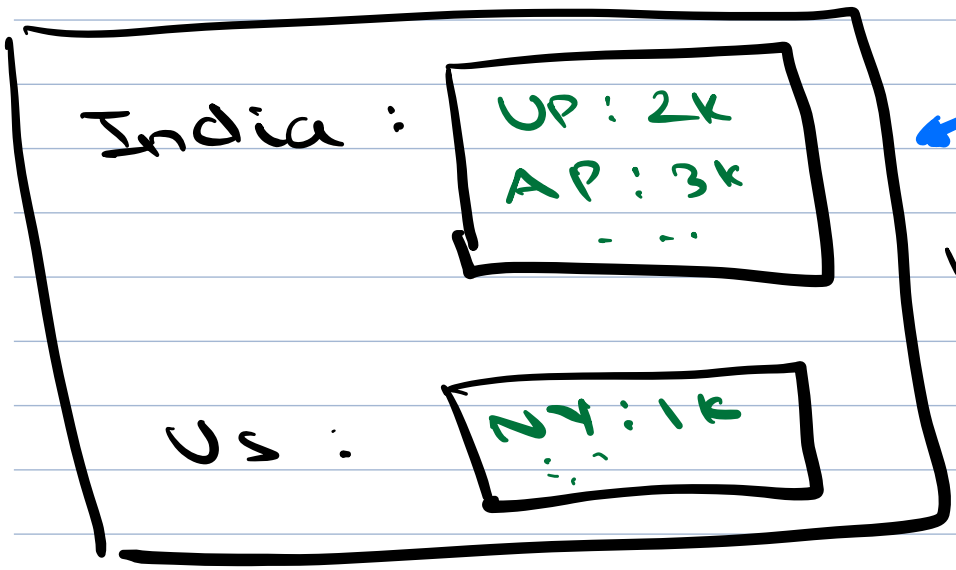
key → Country

value → Population of each state

Hashmap < String , Hashmap < String, Long > > hm

key value
State Population

Hashmap < String, Long >



	Java	C++	Python
Hashmap	Hashmap	unordered_map	dict
Hashset	Hashset	unordered_set	set

JS	C#
map	dictionary
set	hashset

Hashmap <K, V>

Hashset <K>

Ind → 2K
Us → 3K

Hashmap <key, value>

insert <key, value>
search <key>
remove <key>
update <key, new value>

size() → no. of keys in hashmap

All operations in O(1) (avg) for a single operation

Hashset <key>

insert (key)
search (key)
remove (key)

size() → no of keys in hashset

update (key) → remove (key)
↓
insert (new key)

1. Find Frequency of numbers

Given N array elements & Q queries. For each query, find frequency of given element in that query.

N = 10 ar[10] = <2, 6, 3, 8, 2, 8, 2, 3, 8, 10>

Query = 4

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

Idea: For every query, iterate and get count

TC: O(QN)
SC: O(1)

Iterate the array once and store info in a better manner

Frequency of elements

Hashmap <K, V>

array element — frequency

Hashmap <int, int> hm

HM
elem, freq

2, ~~1~~ ~~2~~ 3
6, 1
3, ~~1~~ 2
8, ~~1~~ ~~2~~ 3
10, 1

```
Hashmap <int, int> hm

for (i = 0 ; i < N ; i++) {          → N
    if (hm.search (A[i]) == true) {
        hm [A[i]] ++
    }
    else {
        hm.insert (A[i], 1)
    }
}

for (i = 0 ; i < Q ; i++) {          → Q
    int elem = q[i]
    if (hm.search (elem) == true) {
        print (hm [elem])
    }
    else {
        print (0)
    }
}
```

TC: O(N + Q)

SC: O(N)

10:35

2. Given N array elements, find no. of distinct elements.

Ex 1 ar[5] : <3, 5, 6, 5, 4>

Ex 2 ar[7] : <6, 3, 7, 3, 8, 6, 9>

3.