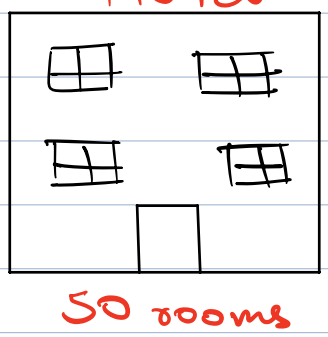


Register

Hash Table

Key (Room No.)	Value (Occupied)
101	✓
102	✗
⋮	⋮
⋮	✓

HashMap < Key , Value >

Key → Unique

Value → Information

Store population of every country.

Key : Country Name → String

Value : Population → Long

HashMap < String, Long > hm;

hm → name

No. of states for every country.

Key : Country Name → String

Value : No. of States → Int

HashMap < String, Int > hm.

for every country, all state names.

KEY: Country Name → String
VALUE: All state Name → List<String>

HashMap<String, List<String>> hm;

C++: vector
Python: List
Java: ArrayList

---

for every country, population of every state.

KEY: Country Name → String
VALUE: HashMap<State, Population>
String Long.

HashMap

KEY (Country Name)	VALUE HashMap<String, Long>
1) India	Andhra — 5000 Rajasthan — 6000 Karnataka — 10,000
2) USA	Nevada 60 Texas 50 CA 70

Search ⇒ hm.get(key) ⇒ O(1)

How ?? (Advanced DSA)

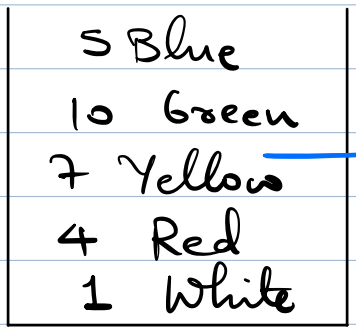
hm.get("India")

⇓

HashMap

hm.get("India").get("Andhra")

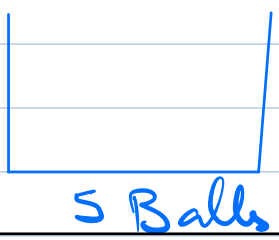
HashSet



→ Balls of different colors

Goal: How many unique colors of balls you have ??

Ans = 5



⇒ HashSet

HashMap <Key, Value>

SIZE : {No. of Keys present in HM}

containsKey (Key)
Return True if the Key is present in HM

insert (Key, value)

delete (Key)

get (Key)
Return the value corresponding to Key

update (Key, Value)

HashSet <Key>

SIZE : {No. of Keys present in HS}

containsKey (Key)
Return True if the Key is present in HS

insert (Key)

delete (Key)

T.C. of all above functions = O(1)

Pseudo Code	Java	C++	Python	JS	C#
HashMap	HashMap	unordered_map	Dictionary	Map	Dictionary
HashSet	HashSet	unordered_set	Set	Set	Hashset

H.W. Search what is the name of above mentioned functions in your language.

Q Given an integer array of size N.

Q queries → K ⇒ Frequency of K in the array.

Const: $1 \le N \le 10^5$, $1 \le Q \le 10^5$, $1 \le A[i] \le 10^9$

A = [2, 6, 3, 8, 2, 8, 2, 3, 8, 10, 6]

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

Soln 1) Brute force

∀ queries : iterate & find frequency (count)

T.C. = O(Q × N)
S.C. = O(1)

2) Optimise

⇒ Precompute the frequency of all array elements.

HashMap < Array Element , Frequency >

A = [2, 6, 3, 8, 2, 8, 2, 3, 8, 10, 6]

< Key , Value >

Key	Value
2	~~1~~ ~~2~~ 3
6	~~1~~ 2
3	~~1~~ 2
8	~~1~~ ~~2~~ 3
10	1

Code

```
HashMap < int, int > hm;

for (i=0; i<N; i++) {
        if ( ! hm.containsKey (A[i]) ) {
                hm.insert (A[i], 1);
        }
        else {
                count = hm.get (A[i]);
                hm.update (A[i], count+1);
        }
```

```
    }

    input (Q);
    for (i = 0; i < Q; i++) {

        input (element);
        if (! hm.containsKey (element)) {
            print (0);
        }
        else {
            print (hm.get (element));
        }
    }
```

T.C. = $O(N+Q)$

S.C. = $O(N)$

Given an integer array of size N. Find the count of distinct elements.

A = [3, 5, 6, 5, 4, 3, 1, 2, 5, 1]

(3, 5, 6, 4, 1, 2) Ans = 6.

Soln Hashset

```
HashSet<int> hs;
for (i=0; i<N; i++) {
        hs.insert(A[i]);
}
return hs.size();
```

T.C. = O(N)
S.C. = O(N)

Check if all elements are distinct or not.

```
if (hs.size() == N) {
        return True;
}
else {
        return false;
}
```

Given an integer array of size N. Find the first non repeating element.
↓ (first element from start)

A = { 3, 2, 4, 1, 3, 2, 5 }

→ Ans = 4

Soln 1) Brute force

∀ elements from 0 → N-1, iterate & check if (freq == 1)

T.C. = $O(N^2)$
S.C. = $O(1)$

2) HashMap

1) Create a freq map.

2) Iterate over HM & find the first element with freq = 1

(HashMap & HashSets are unordered)

⇒ Iterate over the array & check freq

⇒ 1st element with freq = 1 is ans.

T.C. = O(N)
S.C. = O(N)

Doubt

Active Learning → Revision (New Questions)

⇓

Learn & Implement (Ass. H.W.)

Sunday afternoon ⇒ PS session (Optional)

Doubts

⇓

Dashboard