HashMap- [key - Value]

Highest occ. char — Sort array. — nlogn
— freq. array —
— hashmap.

TC of every operation in HM is constant O(1)

key → Value

(String) → (Integer)

"Delhi" → 250

"mumbai" → 30

pair

key

String

Integer

Boolean

Double

character

Value:

String

Integer

Boolean

Double

character

Array list

array

stack

HM

Imp. Points

a = 1]54
a = 4

→ all keys will always be unique.

→ keys are case sensitive (a, A)

→ values can be repeated.

→ if same key is added again, then previous value will be over ridden.

→ data is unorganised (no indexing)

→ load factor 0.76

Syntax

HashMap < Data type of key , DT of value > map = new HashMap <>();

HashMap < String , Integer > map = new HashMap <>();

Inbuilt functions

→ map.put("Delhi", 250); // to add pair in HM

→ map.get("Delhi"); // return value

→ map.remove("Mumbai"); // complete pair will be remove

```
map.size(); // size of HM
map.isEmpty(); // whether HM empty or not.

map.containsKey("Delhi"); // true

map.containsValue(500); // false.
```

TC = O(1)

Integer → String

~~25 → "abc"~~ efg

26 → "xyz"

25 → "efg"

```
map.isEmpty(); // false
map.put() (25, "efg");
map.put(26, "xyz");
map.size(); // 2
map.get(25); // efg
map.containsKey(20); // false
map.containsValue("abc");
// false.
```

HW_Contains Duplicate?

```java
Scanner s = new Scanner(System.in);
int n = s.nextInt();

HashSet<Integer> st = new HashSet<>();

boolean ans = false;

for(int i = 0; i < n; i++){
    int num = s.nextInt();

    if(st.contains(num)){
        ans = true;
        break;
    }
    else{
        st.add(num);
    }
}

if(ans){
    System.out.println("true");
}
else{
    System.out.println("false");
}
}
```

TC → O(n)

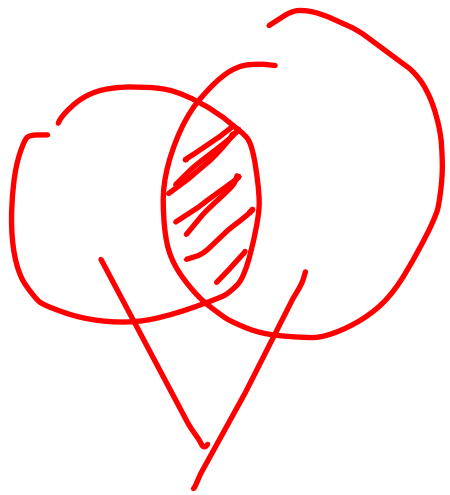
SC → O(1)

HW_First Unique Character in a String

```
Scanner s = new Scanner(System.in);

String str = s.nextLine();

HashMap<Character, Integer> freq = new HashMap<>();

for(char c : str.toCharArray()){
    freq.put(c, freq.getOrDefault(c, 0) + 1); // freq add
}
int index = -1;
for(int i = 0; i < str.length(); i++){
    if(freq.get(str.charAt(i)) == 1){
        index = i;
        break;
    }
}
        System.out.println(index);
}
```

O(n)

HW_Union of two arrays 5



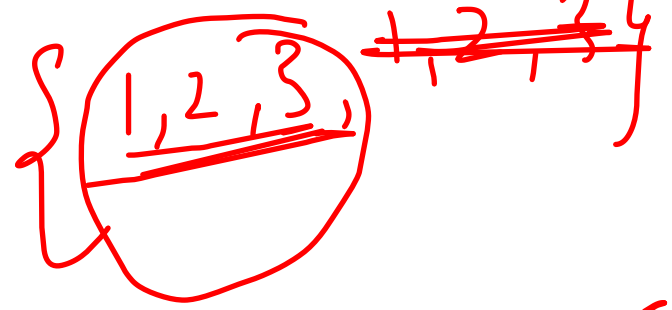
{1,2,3}

{1,2,3}

{1,2,3} ∪ {1,2,3}

{1,2,3} ∪ {4,5}

⇒ {1,2,3,4,5}



a[] = [1,2,3,4,5]

b[] = [1,2,3]

ans[] = [1,2,3,4,5] → 5

O(n) → TC

O(1) → SC

```
    Scanner s = new Scanner(System.in);
    int n = s.nextInt();
    int[] arr1 = new int[n];
    for(int i=0; i < n;i++){
        arr1[i]=s.nextInt();
    }
    int m = s.nextInt();
    int[] arr2 = new int[m];
    for(int i=0; i < m;i++){
        arr2[i]=s.nextInt();
}

    HashSet<Integer> union = new HashSet<>();

    for(int i = 0; i < n; i++){
        union.add(arr1[i]);
    }

    for(int i = 0; i < m; i++){
        union.add(arr2[i]);
    }

    System.out.println(union.size());
}
```

HW_Majority Element 5

3, 2, 3, 4, 1, 1, 5

Map.entry

key	value
3	2
2	1
4	1
1	2
5	1

```java
public static int majority(int arr[]){

    HashMap<Integer, Integer> freq = new HashMap<>();
    for(int num : arr){
    freq.put(num, freq.getOrDefault(num, 0) + 1);
    }

    int count = arr.length/2;
    for(Map.Entry<Integer, Integer> entry : freq.entrySet()){
        if(entry.getValue() > count){
            return entry.getKey();
        }
    }
    return -1; // will never enter
}
```

O(n)