⇒ Hashmap [time complexity of every operation in HM is constant]

T.C = O(1)

Key → Value
(String) (Integer)

Hashmap

pairs

"Banglore" → 250
"Delhi" → 200
"Chennai" → 350
"Mumbai" → 300
"Rajasthan" → 240
"banglore" → 200

Key
String
Integer
Boolean
Double
Character
⋮

Value
String
Integer
Boolean
Double
Character
⋮
ArrayList
array
Stack
HM

Imp. points to note :-

1) all keys are always unique
2) keys are case sensitive
3) values can be repeated
4) if same key is added again, then previous one will be overrided.
5) data is unorganised. (no indexing)

Syntax

HashMap< DataType of Key, DataType of Value > map = new HashMap<>();

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

Inbuilt f^n

- map.put("Banglore", 250); // to add pair in HM
- map.get("Chennai"); // return value paired with key
  (return typ :- data type of value)
- map.remove("Rajasthan"); // entire pair will be removed

(Evergreen f^n :- map.size() & map.isEmpty())

↳ map.containsKey("Hyd"); // false

↳ map.containsValue(200); // true

Note:- all are of O(1) T.C

Integer → String

map

25 → "xyz"
26 → "efg"

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

Word Meaning

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    HashMap<String, String> map = new HashMap<>();
    while (true) {
        int n = scn.nextInt();
        if ( n == 1 ) {
            String word = scn.next();
            String meaning = scn.next();
            map.put(word, meaning);
        } else if (n == 2) {
            String word = scn.next();
            if ( map.containsKey(word) == true ) {
                System.out.println(map.get(word));
            } else {
                System.out.println("-1");
            }
        } else if (n == 3) {
            String word = scn.next();
            map.remove(word);
        } else if (n == 4) {
            return;
        }
    }
}
```

T.C = O(n)

S.C = O(n)

Same Number Same Frequency

$n = 8$

arr = [3, -2, 7, -2, 7, 3, 3, 1]

(|key| == value)

(num) (freq)

Integer → Integer

map

3 → 3

-2 → 2

7 → 2

1 → 1

1) loop from start to end in array

1.1) if curr ele is present

int freq = map.get(curr no.)

map.put(curr, freq+1);

1.2) else

map.put(curr, 1);

ans = 1, -2, 3

⇒ How to traverse in HashMap

only using for each loop

Syntax

```
for( Map.Entry<Integer, Integer> e : map.entrySet()){
    Integer key = e.getKey();
    Integer value = e.getValue();
}
```

Code

```java
public static void sameNumSamFreq(int[] arr, int n) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        if ( map.containsKey(arr[i]) ) {
            int freq = map.get(arr[i]);
            map.put( arr[i], freq + 1 );
        } else {
            map.put( arr[i], 1 );
        }
    }

    ArrayList<Integer> ans = new ArrayList<>();
    for (Map.Entry<Integer, Integer> entry : map.entrySet()) {
        int key = entry.getKey();
        int value = entry.getValue();

        if ( Math.abs(key) == value ) {
            ans.add(key);
        }
    }

    Collections.sort(ans);
    for (int i : ans) {
        System.out.println(i);
    }
}
```

$T.C = O(n)$

Character and it's Frequency

n =

```
6
a b a d b c
```

Character → Integer

a → ~~1~~ 2

b → ~~1~~ 2

d → 1

c → 1

Code

```java
public static void charByFreq(char[] arr, int n) {
    HashMap<Character, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        if ( map.containsKey(arr[i]) ) {
            int freq = map.get(arr[i]);
            map.put( arr[i], freq + 1 );
        } else {
            map.put( arr[i], 1 );
        }
    }

    ArrayList<Character> ans = new ArrayList<>();
    for (Map.Entry<Character, Integer> entry : map.entrySet()) {
        char key = entry.getKey();
        int value = entry.getValue();
        ans.add(key);
    }

    Collections.sort(ans);
    for (char i : ans) {
        System.out.println( i + " " + map.get(i) );
    }
}
```

T.C=O(n)