

# Two Sum 14

arr

0	1	2	3
2	7	11	15

target = 9

↑   ↑   ↑   ↑

$$\text{num1} + \text{num2} == \text{target}$$
$$\text{num2} == \text{target} - \text{num1}$$

map.get(num1)  
map.get(num2)

0, 1
1, 0

map

Integer, Integer

2 → 0

7 → 1

11 → 2

15 → 3

target = 9

num1 = 2, num2 = 7

num1 = 7, num2 = 2

num1 = 11, num2 = -2

num1 = 15, num2 = -6

Code

T.C =  $O(N)$

S.C =  $O(N)$

```
public static void twoSum(int[] arr, int n, int target) {  
    HashMap<Integer, Integer> map = new HashMap<>();  
    for (int i = 0; i < n; i++) {  
        map.put( arr[i], i );  
    }
```

$O(N)$

$O(N)$

```
        int[] answer = new int[2];  
        for (int i = 0; i < n; i++) {  
            int num1 = arr[i];  
            int num2 = target - num1;  
            if ( map.containsKey( num2 ) ) {  
                if ( i != map.get( num2 ) ) {  
                    answer[0] = i;  
                    answer[1] = map.get(num2);  
                    Arrays.sort(answer);  
                    System.out.println(answer[0] + " " + answer[1]);  
                    return;  
                }  
            }  
        }  
    }  
}
```

# Unique Number of Occurrences

arr      1      3      3      1      1      1      2      2      2  
          ↑      ↑      ↑      ↑      ↑      ↑      ↑      ↑      ↑

map

val, freq

1 → 4

3 → 2

2 → 3

= =

set

4

2

3

map

1 → 3

2 → 4

5 → 3

4 → 1

size = 4

!=

set

3

4

1

size = 3

set.add(val)  
set.remove(val)  
set.size();  
set.contains(3)

code

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    System.out.println(uniqueOcc(arr, n));
}

public static boolean uniqueOcc(int[] arr, int n) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        if ( !map.containsKey(arr[i]) ) {
            map.put( arr[i], 1 );
        } else {
            map.put( arr[i], map.get(arr[i]) + 1 );
        }
    }

    HashSet<Integer> set = new HashSet<>();
    for (int i : map.values()) {
        set.add(i);
    }

    if ( map.size() == set.size() ) return true;
    else return false;
}
```

$T.C = O(N)$

$S.C = O(N)$

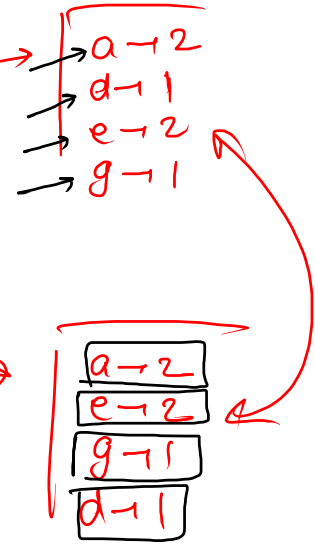
# Valid Anagram 5

str1 = "adeega"

str2 = "dgeaea"

str1  
map

a	→	2
d	→	1
e	→	2
g	→	1



code

```
public static boolean validAnagram(String str1, String str2) {  
    HashMap<Character, Integer> map1 = new HashMap<>();  
    for (int i = 0; i < str1.length(); i++) {  
        if ( !map1.containsKey( str1.charAt(i) ) ) {  
            map1.put( str1.charAt(i), 1 );  
        } else {  
            map1.put( str1.charAt(i), map1.get( str1.charAt(i) ) + 1 );  
        }  
    }  
}
```

```
    HashMap<Character, Integer> map2 = new HashMap<>();  
    for (int i = 0; i < str2.length(); i++) {  
        if ( !map2.containsKey( str2.charAt(i) ) ) {  
            map2.put( str2.charAt(i), 1 );  
        } else {  
            map2.put( str2.charAt(i), map2.get( str2.charAt(i) ) + 1 );  
        }  
    }  
}
```

```
→ for (Map.Entry<Character, Integer> e : map1.entrySet()) {  
    char ch = e.getKey();  
    int freq = e.getValue();  
    if ( !map2.containsKey(ch) ) {  
        return false;  
    }  
    if ( map2.get( ch ) != map1.get( ch ) ) {  
        return false; ←  
    }  
}  
return true;
```

map1

a	→	2
b	→	3
c	→	1

ch = ~~a~~ ~~b~~ c  
freq = ~~2~~ ~~3~~ 1

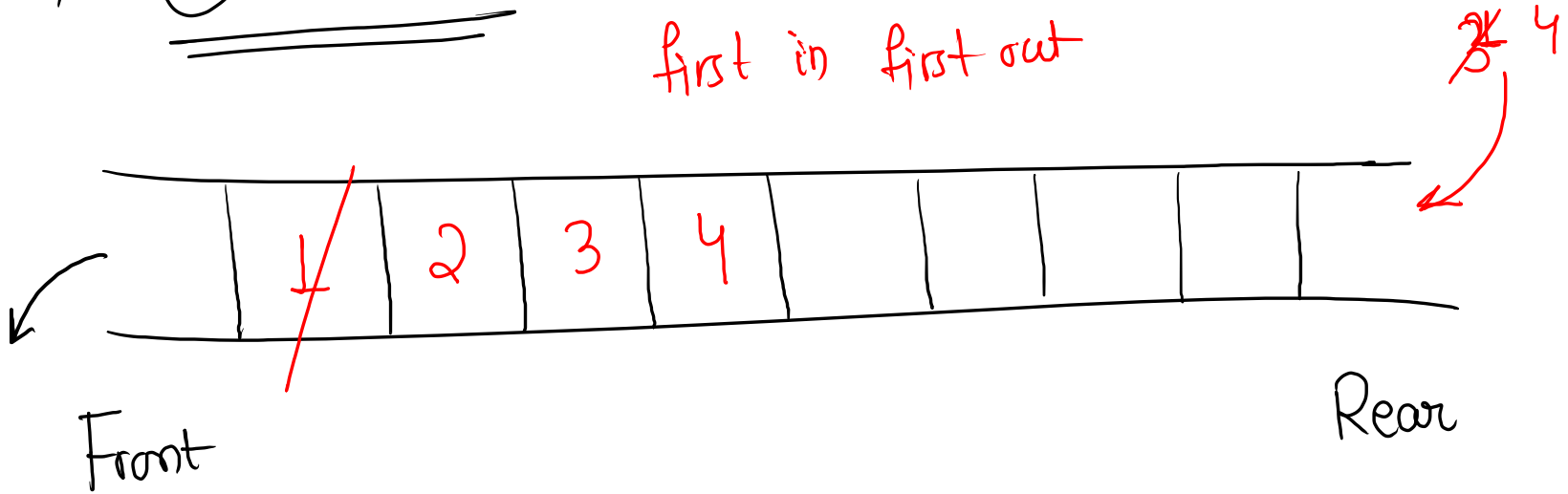
map2

b	→	3
a	→	2
c	→	2

⇒ Queue

(FIFO)

first in first out



always add elements from rear  
and remove elements from front

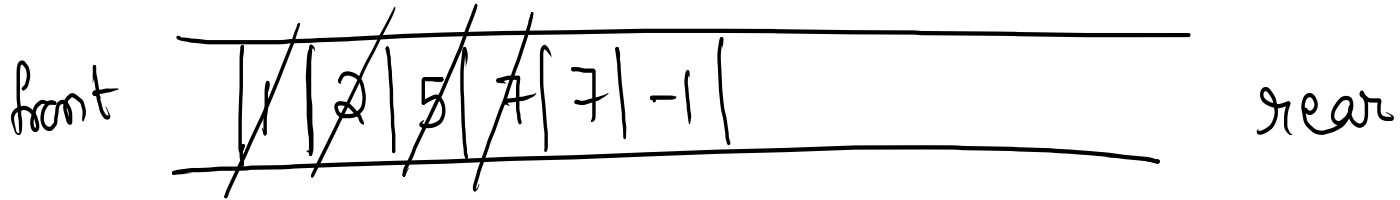
## Syntax:-

Queue < Integer > que = new LinkedList<>();

## Inbuilt functions

- ↳ que.add(x); // to add an element in rear
- ↳ que.remove(); // to remove element
- ↳ que.poll(); // from front
- ↳ que.peek(); // return value at front
- ↳ que.size(), que.isEmpty();





que.add(1)  
 add(2)  
 add(5)  
 add(7)  
 1  
 → que.peek() → 1  
 print(que.poll()) → 1  
 print(que.poll()) → 2  
 que.add(7)  
 que.add(-1)

que.peek() → 5  
 print(que.poll()) → 5  
 (que.poll()) → 7