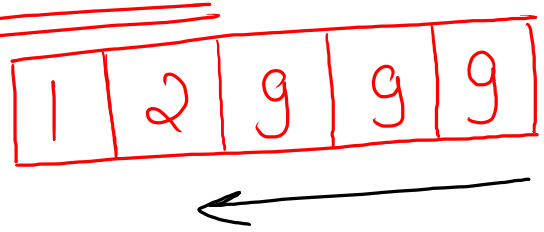
Questions

- Four sum question
- Product except itself ✓
- three sum
- desired string
- Reverse by words
- majority elements
- prefix sum b/w L and R

- Theory of 1D array
  - syntex
- Questions
  - maximum of array ✓✓
  - Plus one in an array
  - Second largest element

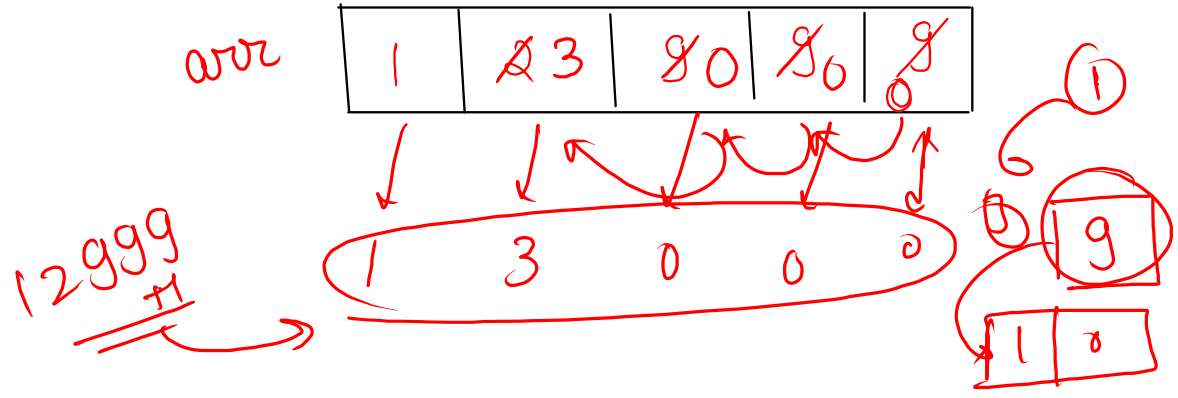
↳ Plus one

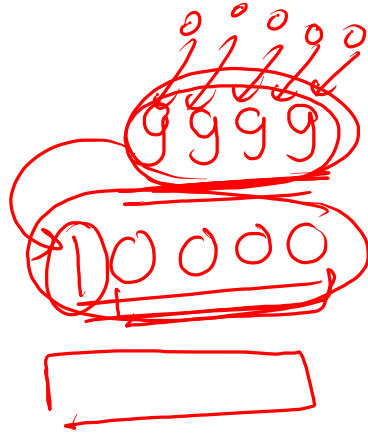


concept

number could be either

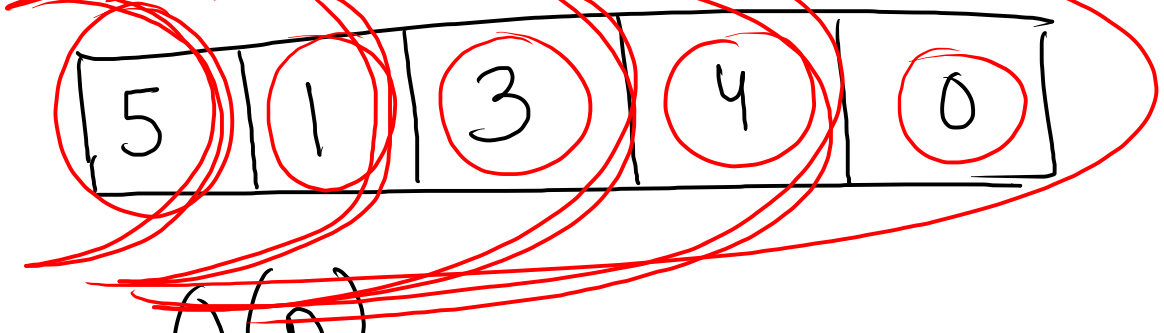
or






```java
public static int[] plusOne(int[] arr, int n) {
    for (int i = n - 1; i >= 0; i--) {
        if ( arr[i] < 9 ) {
            arr[i] = arr[i] + 1;
            return arr;
        }
        arr[i] = 0;
    }
    int[] ans = new int[n + 1];
    ans[0] = 1;
    return ans;
}
```

2nd largest



O(n)

largest

2nd largest

1st (update 2nd largest by largest & largest by current ele)

2nd (update 2nd largest only)

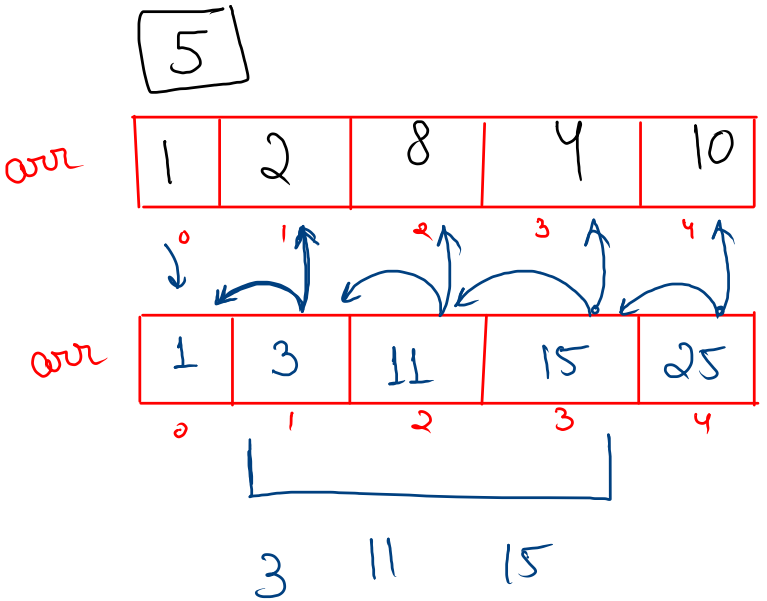
3rd (do nothing)

O(n)

largest = ~~-∞~~ 5

2nd largest = ~~-∞~~ ~~-∞~~ 1 3 4

Print Prefix Sum between L and R



left = 1
right = 3

(prefix sum array)

```
for(int i=1; i<n; i++){
    arr[i] = arr[i] + arr[i-1];
}
```

prefix sum is when all indexes have sum of all the previous values elements

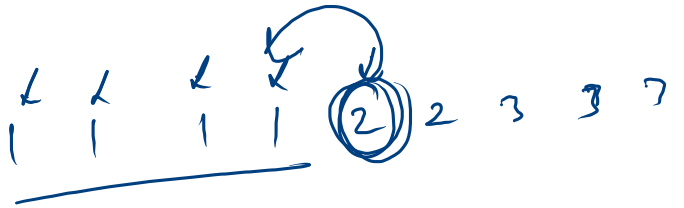
Ques max Count 3

```java
// main logic
public static int maxCount(int[] arr, int n) {
    Arrays.sort(arr);

    int ans = arr[0];
    int maxCount = 1;
    int count = 1;
    for (int i = 1; i < n; i++) {
        if ( arr[i] == arr[i - 1] ) {
            count++;
        } else {
            count = 1;
        }

        if (maxCount < count) {
            maxCount = count;
            ans = arr[i - 1];  // x
        }
    }
    return ans;
}
```

Imp

storing the number 'x'

how many time a number 'x' is appearing

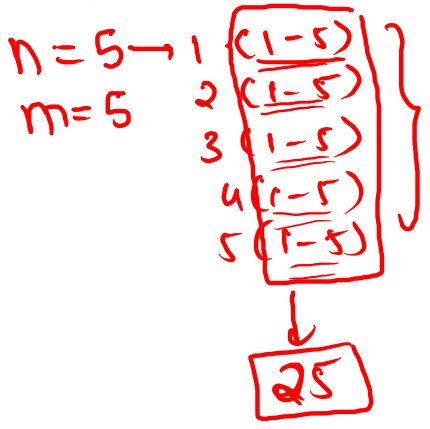


C = ~~1~~ ~~2~~ ~~3~~ 4

ans = 1

Ques double occurence

```java
// main logic
public static void doubleOccurence(int[] arr1, int n, int[] arr2, int m) {
    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = 0; j < m; j++) {
            if ( arr1[i] == arr2[j] ) {
                count++;
            }
        }

        if (count == 2) {
            System.out.print(arr1[i] + " ");
        }
    }
}
```

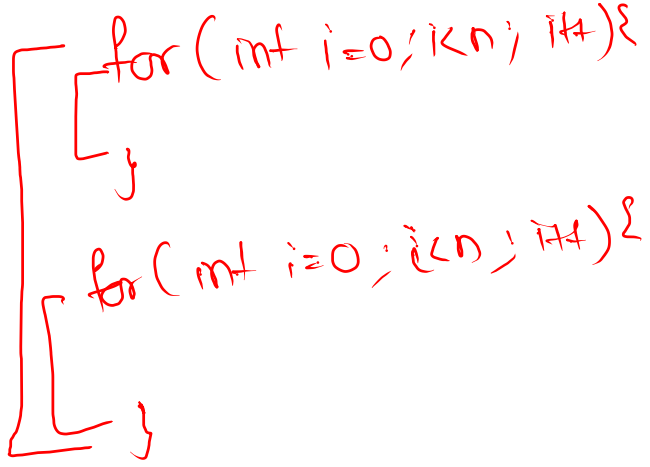


$O(2 \times n) \cong O(n)$

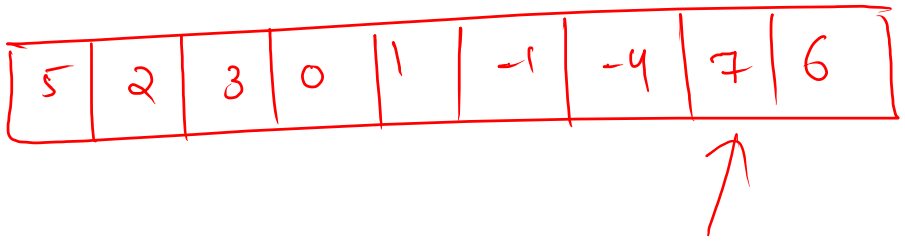
we visiting all elements twice in different loops

$T.C = O(N^2)$ $5 \times 5 = 25$

because for all the elements, I am visiting all the elements again



Kth Largest Element in an Array



asc. order =

desc. order =

HW_k frequent elements

1 1 1 2 2 3 3

number → freq of that no.

1 → 3 ✓

2 → 2 ✓

3 → 2 ✓

k=2 ✓

Consider highest freq element first → 1, 3

then if freq is same then, consider numbers in descending order

print no. only and not freq

(lambda expression)

or

(array as hashmap)

k=3

1, 3, 2

3 Sum

```
6
-2 0 2 4 -2 -8
```

$2 + 3 + (-5) = 0$ _ _

$x + y + z = 0$ —(1)

$x + y = -z$ —(1)

$tar = (-z)$

target sum

arrays.sort() : $-8 \quad -2 \quad -2 \quad 0 \quad 2 \quad 4$

tar

target = $-arr[0]$ $(-z)$
$= -(-8) = 8$ (no ans)

target = $-arr[1]$
$= -(-2) = 2 \Rightarrow (-2, 4)$
$(0, 2)$

ans $(-2, -2, 4)$
$(-2, 0, 2)$
$(-2, 0, 2)$

target = $-arr[2]$
$= -(-2) = 2 \rightarrow (0, 2)$

target = $0 \rightarrow$ no ans

3 Sum Code

```java
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();
    }

    Arrays.sort(arr);
    for (int i = 0; i < n; i++) {
        int target = arr[i] * -1;
        int si = i + 1, ei = n - 1;
        while (si < ei) {
            int sum = arr[si] + arr[ei];
            if (sum < target) {
                si++;
            } else if (sum > target) {
                ei--;
            } else {
                System.out.println(arr[i] + " " + arr[si] + " " + arr[ei]);
                si++;
                ei--;
            }
        }

        while (i + 1 < arr.length && arr[i] == arr[i + 1]) i++;
    }
}
```