Day 5 :- Array

H.W. Find out the 2nd largest number & 2nd smallest no from given number array.

→

0	1	2	3	4
12	24	5	36	31

Brute Force Approach → array → sorted → last → largest ele
small → first ele

5	12	24	31	36
0	1	2	3	4

Bubble Sort → n^2
Insertion → n^2
Selection → n^2
merge — $O(n \log n)$
Quick Sort →

```
void findele( int arr[], int n)
{
    for ( int i= ø ; i<n ; i++)
                 1
    {
        if ( largest_ele < arr[i])
        {
            largest_ele = arr[i]
        }
}
```

→ int largest_ele = arr[0];

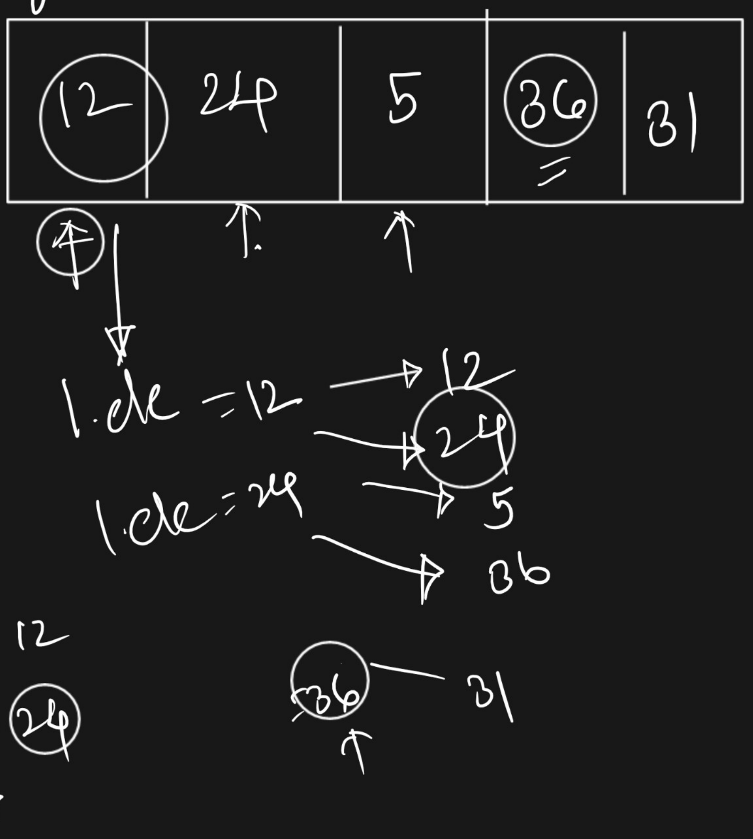
12	24	5	36	31	
0	1	2	3	4	5

largest_ele = ~~12~~ ~~24~~ 36

2) Approach

```
}        S.o.p. ( largest_ele );
```

2nd largest Element



Brute Force :-   Sorted ⇒ $O(n^2)$
↓
last index ⇒ 2nd largest approach

$O(n)$

```
for( int i=0 ; i<n ; i++)
{
    if( s_largest_ele < arr[i] && arr[i] != largest_ele)
    {
        s.largest_ele = arr[i]
```

```
void    resultvalues (int arr[], int n)
{                                                      -2147483648
     int l_ele = Integer.MIN_VALUE;          =>
     int second_l_ele = Integer.MIN_VALUE;   =>  -2147483648
     int s_ele = Integer.MAX_VALUE;          =>  2147483648
     int s_s_ele = Integer.MAX_VALUE;        =>  2147483648


     for(int i=0; i<n; i++)
     {
  36 <-  l_ele = Math.MAX( l_ele, arr[i]);
o(n)
  5 <-   s_ele = Math. ~~MAX~~ MIN ( s_ele, arr[i])
                               2148
      }
     for(int i=0; i<n; i++)
      {   if( s_l_ele < arr[i] && arr[i] != l_ele)
o(n)        {
                s_l_ele = arr[i];

            }
```

	l_el		arr[i]
i=0	-2147	i=0	12
i=1	12	i=1	24
i=2	24	i=2	5
i=3	24	i=3	36
i=4	36	i=4	31
	$\boxed{36}$		

```
        if ( s_s_ele > arr[i] && arr[i] != s_ele)
        {
              s_s_ele = arr[i]
        }
    }

    s.o.p. (s_l_ele);
    s.o.p (s_s_ele);
}
```

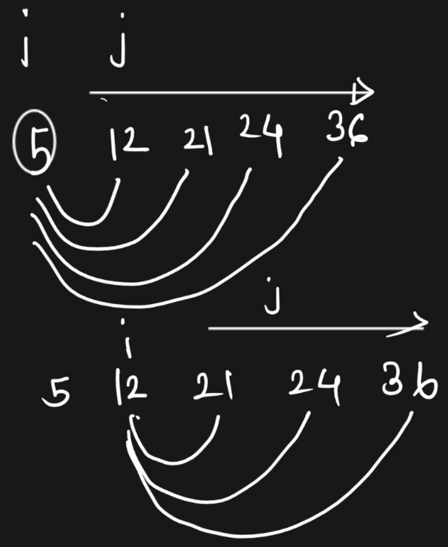
Time Complexity $= O(n) + O(n)$
$= 2O(n)$
$\simeq \underline{O(n)}$ ↑

H.W. ↑ 2nd largest & 2nd smallest ele from array

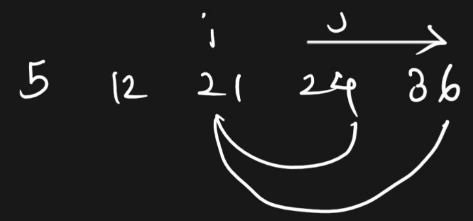
② Find out given array is sorted or not :-

5	12	21	24	36
0	1	2	3	4

Brute Force :-



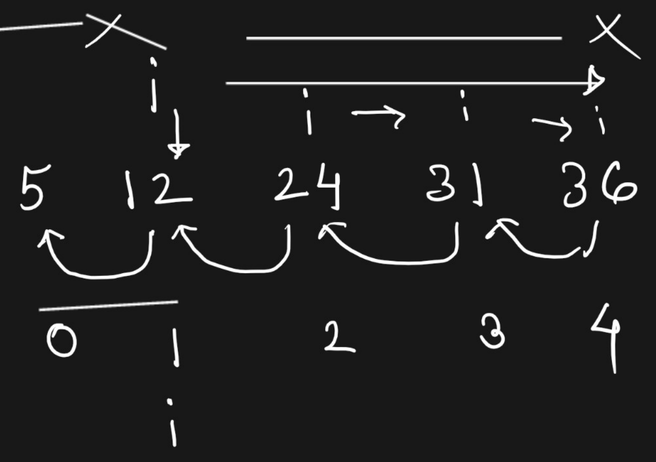
```
for ( int i=0 ; i<n ;  i++)  → O(n)
{
    for ( int j=1 ; j<n ; j++) → O(n)
    {
        if ( arr[i] > arr[j] )
            return false;
    }
}
return true;
```



Time Complexity

$= O(n^2)$

2nd Approach :-



$arr[i] = 12$

$arr[i-1] = 5$

$i=1$	$i=2$	$i=3$	$i=4$
$i-1=0$	$i-1=1$	$i-1=2$	$i-1=3$

$i = 1$

$i-1 = 1-1 = 0$

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

$\Bigg\}$ Time Complexity $O(n)$

③ Reverse the array element :-

12	5	30	24	55

5

O/p 55 , 24 , 30 , 5 , 12

12 , 5 , 30 , 24 , 55

0 1 2 3 4

```
for ( int i=0 ; i<n ; i++)
{
      S.o.p. (arr[i]);
```

```
for ( int i=n-1 ; i>=0 ; i--)
{     S.o.p. (arr[i]);      }
```

$i=0 \rightarrow arr[0]$

$i=1 \rightarrow arr[1]$

$i=2 \rightarrow arr[2]$

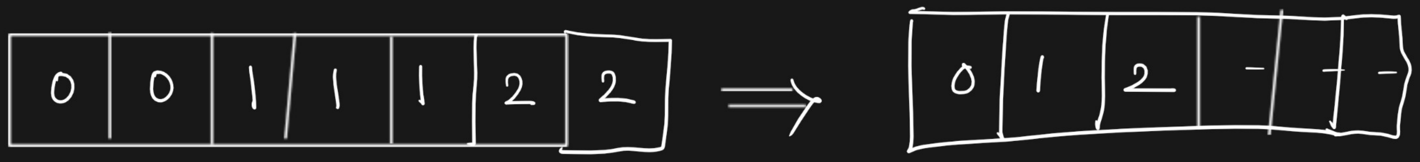
$i=3 \rightarrow arr[3]$

$i=4 \quad arr[4]$

$i=5$

$5<5$ ✗

④ LeetCode Pbm No. 26 :- Remove Duplicate Entries from Array :-



Maintain the relative order

0 1 2 ✓

2 0 1 X

1 0 2 X

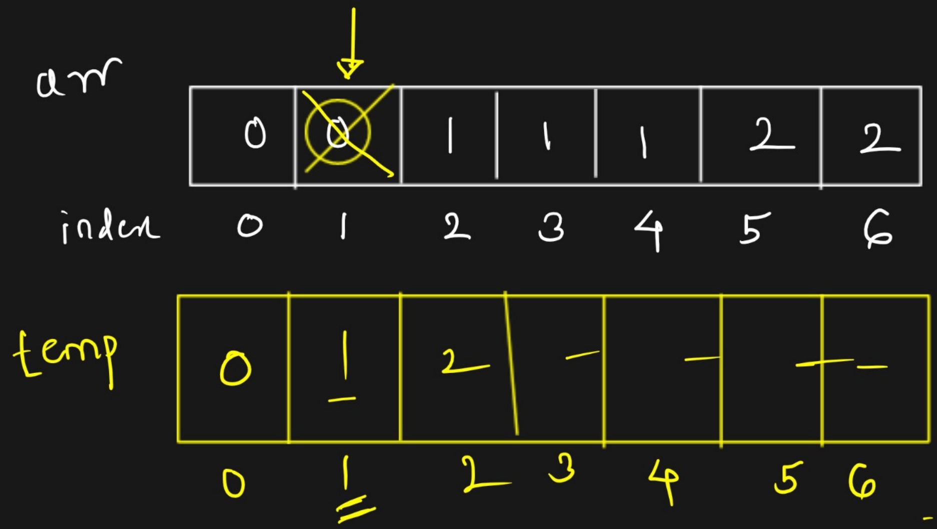
o/p ⇓

no. of unique ele k = 3

```
size = arr.length;

int temp[] = new int[size];

temp[0] = arr[0];
int k = 1;
for(int i=1; i<n; i++)      (n ← size)
{
    if( arr[i]==arr[i-1])
        continue;
```



i=1	i=2	i=3	i=4	i=5	i=6
0	1	1	1	arr[5]=2	arr[6]=2
i-1=0	i-1=0	i-1=2	i-1=3	i-1=4	i-1=5
0		arr(2)=1	arr[3]=1	arr[4]=1	arr[5]=2

0 = 0
continue

1 ≠ 0
else
temp[k] = 1
k++ => 2

1 = 1
skip

1 = 1
skip

2 ≠ 1
else
tem[2] = 2
k++
2++ = 3

2 = 2
skip

```
else
{
    temp [k] = arr[i];
    k++;
}
}
```

0 → 3

```
for(int i = 0; i<k; i++)
{
    arr[i] = temp[i];
}

return k;
```

H.W:- 1) Leetcode Pbm No:- 169

2) Find out even no from given integer array

3) —"— odd no

4) prime no → return multiple count

5) Find out no from array that fully divisible by 5.

6) Find out given array is palindrome const

7) Find out summation of all given array element & it's average.

8)