PAGE No.
DATE / /

12/04/25

Arrays problems:

Q1. Largest element in the array.

arr[] = {2,5,1,3,0}
output = 5

arr = {8,10,5,7,9}
output = 10

2	5	1	3	0

Approach-1
Sorting
but: time: O(Nlog

So, best approach

l ↓ i ——→

arr	2	5	1	3	0
	0	1	2	3	4

- largest = arr[0]
- i → 1 ——→ n-1

i	largest	a[i]	is a[i] > largest so, ~~a[i] =~~ || largest=a[i
1	2	5	5
2	5	1	5
3	5	3	5
4	5	0	5

loop ends: largest = 5.

Pseudocode:

```
largest = arr[0]
for (i=1; i<n~~-1~~; i++)
        if (arr[i] > largest)
                largest = arr[i]
    return largest.
```

time : O(N)
space : O(1)

Q2 find second largest and second smallest

eg: [1,2,4,7,7,5]	eg: [1]
second smallest = 2	" " = -1
second largest = 5	" " = -1

Brute approach

- find smallest, then find number just greater. ⟶ second smallest.
- find largest, then find number just smaller ⟶ second largest.

0	1	2	3	4	5
1	2	4	7	7	5

i Small sec small large sec large

i	arr[i]	1	0	1	1
		2	1	1	2
		4	2	1	4
		7	3	1	7.
		7	4	1	7
		5	5	1	7

small = min(small, arr(i))
large = max (large, arr(i))

Small	large
(1,2)1	(1,2)2
(1,4)1	(2,4)4
(1,7) 1	(4,7)7.
(1,7) 1	(7,7)7
(1,5) 1	(7,5)7

.first traversal:
small = 1
large = 7.
↓
next step ignore this, and Check again

in next iteration: first (smallest, largest) will get removed

In Pseudocode:

```
small
larges'
sec_small , sec_large
i                                          O(N)
for(i=0; i<n; i++){
        ~~if~~ small = math.min(small,arr[i]
           large = math.max(large,arr[i]
        }
for(i=0; i<n; i++)
      if(arr[i] < ~~sec~~ sec_small && arr[i]!=sma
O(N)              sec_small = arr[i]

      if(arr[i] > sec_larg && arr[i]!=largest)
                  sec_largest = arr[i]
   }
```

time: O(2n) ≈ O(N)
space: O(1)

Q3. Check if array is sorted.

Q. [1,2,3,4,5]
output = true

[5,4,6,7,8]
output : false

Code:

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

return true.
```

time : O(N)
space : O(1)

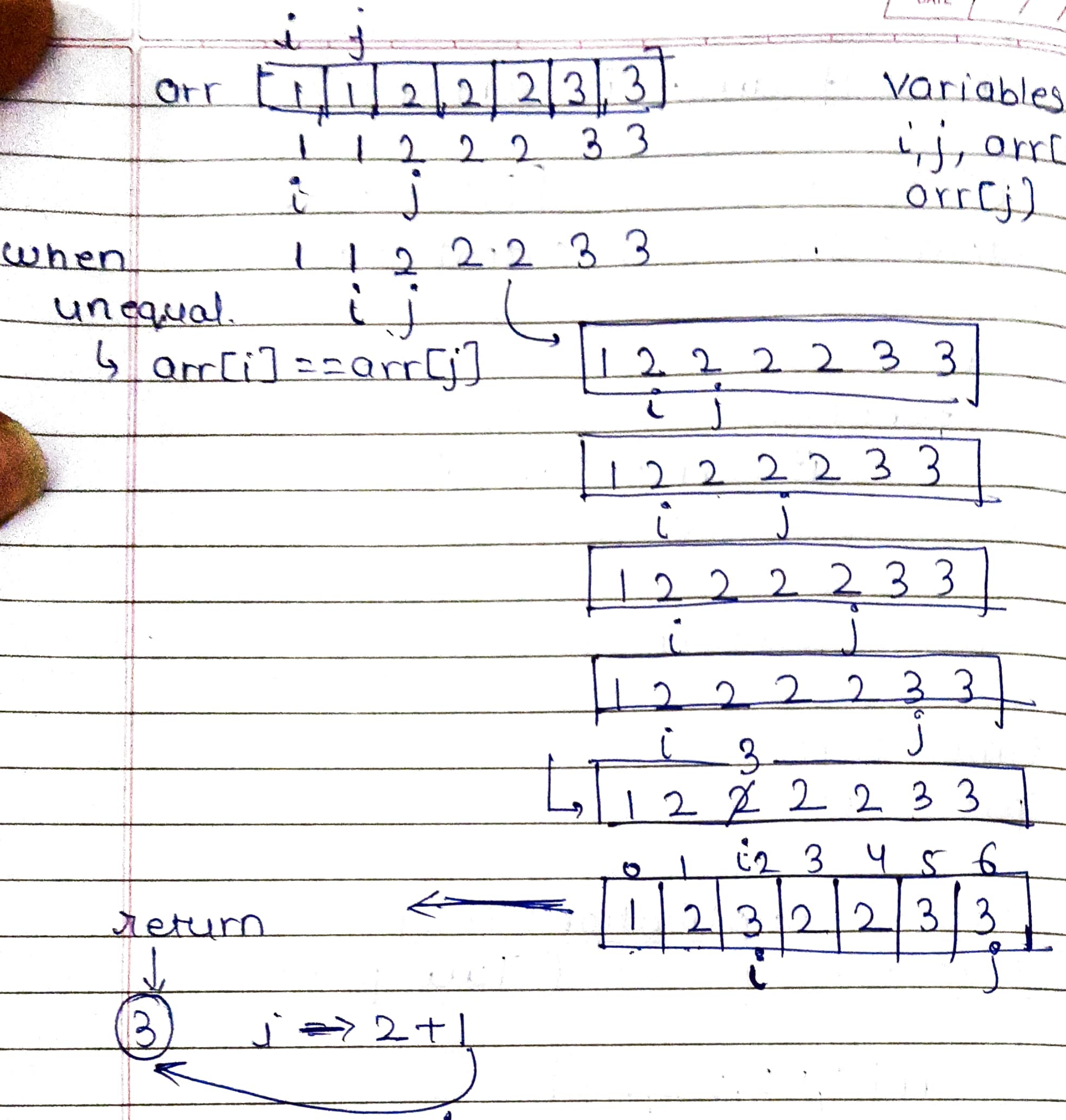
Q4. Remove Duplications in-place from Sorted array. (Two pointers)

eg arr [1,1,2,2,2,3,3]
arr [1,2,3,_,_,_,_] output = 3.

eg arr [1,1,1,2,2,3,3,3,3,4,4]
arr [1,2,3,4, ...] output = 4.

Best approach : two pointers.

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



so, return i+1

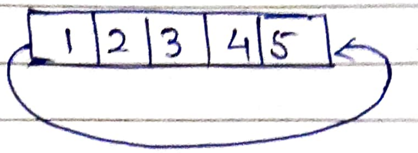
pseudocode

```
removeDuplicates(int[]arr)
    i = 0
    for(j = 1 ; j<n ; j++)
        if(arr[i]! = arr[j])
            i++
            arr[i] = arr[j]
        }
    }
    return i+1
```

time : O(N)
Space : O(1)

Q5. left Rotate the Array by one

given array =



=

=> | 2 | 3 | 4 | 5 | 1 |

- brute:

	0	1	2	3	4	← nums
	1	2	3	4	5	← test
new	2	3	4	5		

+1

i

test[0] = nums[1]
test[1] = nums[2]
test[2] = nums[3]

test[K] = nums(i)
tem[i-1] = arr[i]
tem[n-1] = arr[0]

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

  temp[i-1] = arr[0]
}
```

time : O(n))
Space : ~~O~~ O(N)