TCS | NQT

NATIONAL QUALIFIER TEST

PROGRAMMING LECTURE 1

CS

TCS NQT

(10th) Mon → 7PM onwards Coding Sessions

(11th) Tue → "

(12th) Wed → "

(13th) Thu → ~~Aptitude Session~~

Pattern :-

1) Problem Statement ✓

2) Sample Test Case ✓

3) Approach
4) Code

} on your own

5) Final Code / Approach : (Line)

1) Gate CS 2019 → (4th year BTech)
→ AIR 60
(1st attempt)

2) Amazon (6 months internship)

3) IIT Bombay (2019-2021)

4) Dream → Data Scientist

10K+

Gate CS

Telegram : { Aditya Jain Sir PW }

↓

Share your code here

[t.me/AdityaSir_PW]

[NAT]

#Q. Sort Colors

Problem Statement : Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

Input: nums = [2,0,2,1,1,0]

Output: [0,0,1,1,2,2]

Example 2:

Input: nums = [2,0,1]

Output: [0,1,2]

Constraints:

- n == nums.length
- 1 <= n <= 300
- nums[i] is either 0, 1, or 2

Red → 0

White → 1

Blue → 2.

0 0 1 1 2 2

Appr1: Sorting [2, 1, 0, 1, 0, 2]

↓ Sort in asc ordn

[0, 0, 1, 1, 2, 2]

Best WC: O(nlogn)

Appr2:

Two pointer:

i, j

int n = a.size()

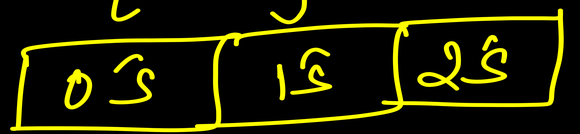
```
for (elem : a)
  {     if(elem == 0)
            i++
        else if (elem == 1)
            j++
  }
```

$i \longrightarrow$ 0's

$j \longrightarrow$ 1's

$(n-i-j) \longrightarrow$ 2's

```
for (k = 0 to i )
{
}
```



[NAT]

#Q. Remove duplicates from Sorted Array

Problem Statement: Given a sorted array arr[] of size n, the goal is to rearrange the array so that all distinct elements appear at the beginning in sorted order. Additionally, return the length of this distinct sorted subarray.

Note: The elements after the distinct ones can be in any order and hold any value, as they don't affect the result.

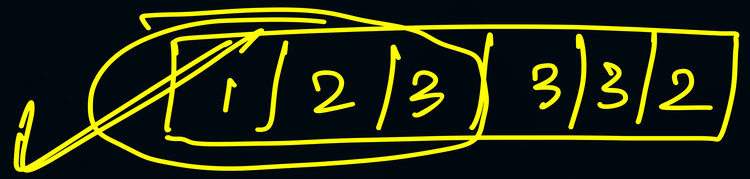
Examples

Example 1:

Input: arr[] = [2, 2, 2, 2, 2]

Output: [2]

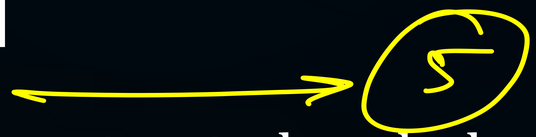
Explanation: All the elements are 2, so only one instance of 2 is kept.



Example 2:

Input: arr[] = [1, 2, 2, 3, 4, 4, 4, 5, 5]

Output: [1, 2, 3, 4, 5]

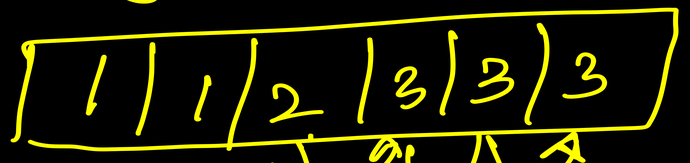


Explanation: All distinct elements are extracted and placed at the beginning in sorted order.

Approach :- Map/Set → Auxiliary Space

Additional Space.

Array is sorted



j = 0

1 2

```
if (a(i) != a(i-1))
{
    a(j) = a(i)
    j++;
}
```

Example 3:

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

Output: [1, 2, 3]

Explanation: No change is required as all elements are already distinct.

[MCQ]

V·V·Imp

#Q. Leaders in an array

Problem Statement : Given an array arr[] of size n, the task is to find all the Leaders in the array. An element is a Leader if it is greater than or equal to all the elements to its right side.

Note: The rightmost element is always a leader.

Examples

Example 1:

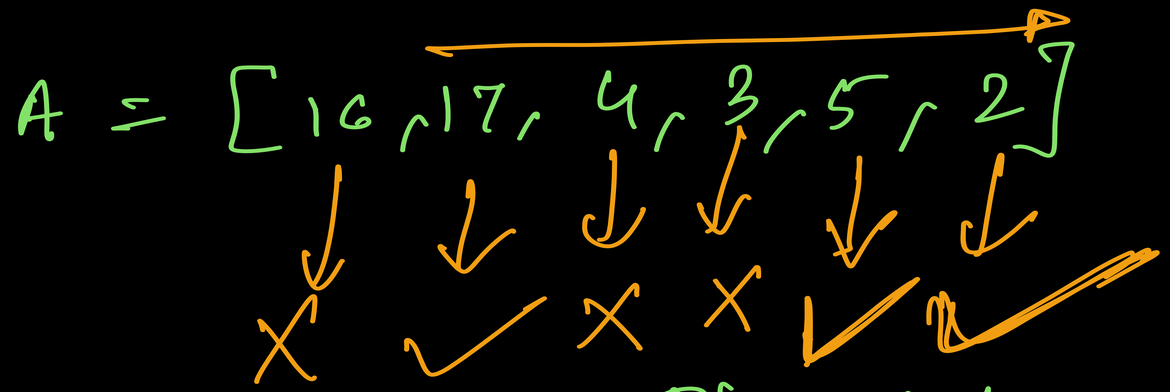
Input: arr[] = [16, 17, 4, 3, 5, 2]

Output: [17, 5, 2]

Explanation:

- 17 is greater than all the elements to its right [4, 3, 5, 2], so it is a leader.
- 5 is greater than all the elements to its right [2], so it is a leader.
- 2 has no elements to its right, so it is a leader.

Telegram: Aditya Jain Sir PW




```
flag = 1;
for ( i = 0 ⟶ n )
        for (j : i+1 ⟶ n)
                if ( a(j) > a(i))
                        flag = 0
                { break;
if( flag == 1 ) ⟶ print a(i)
```

Example 2:

Input: arr[] = [1, 2, 3, 4, 5, 2]

Output: [5, 2]

Explanation:

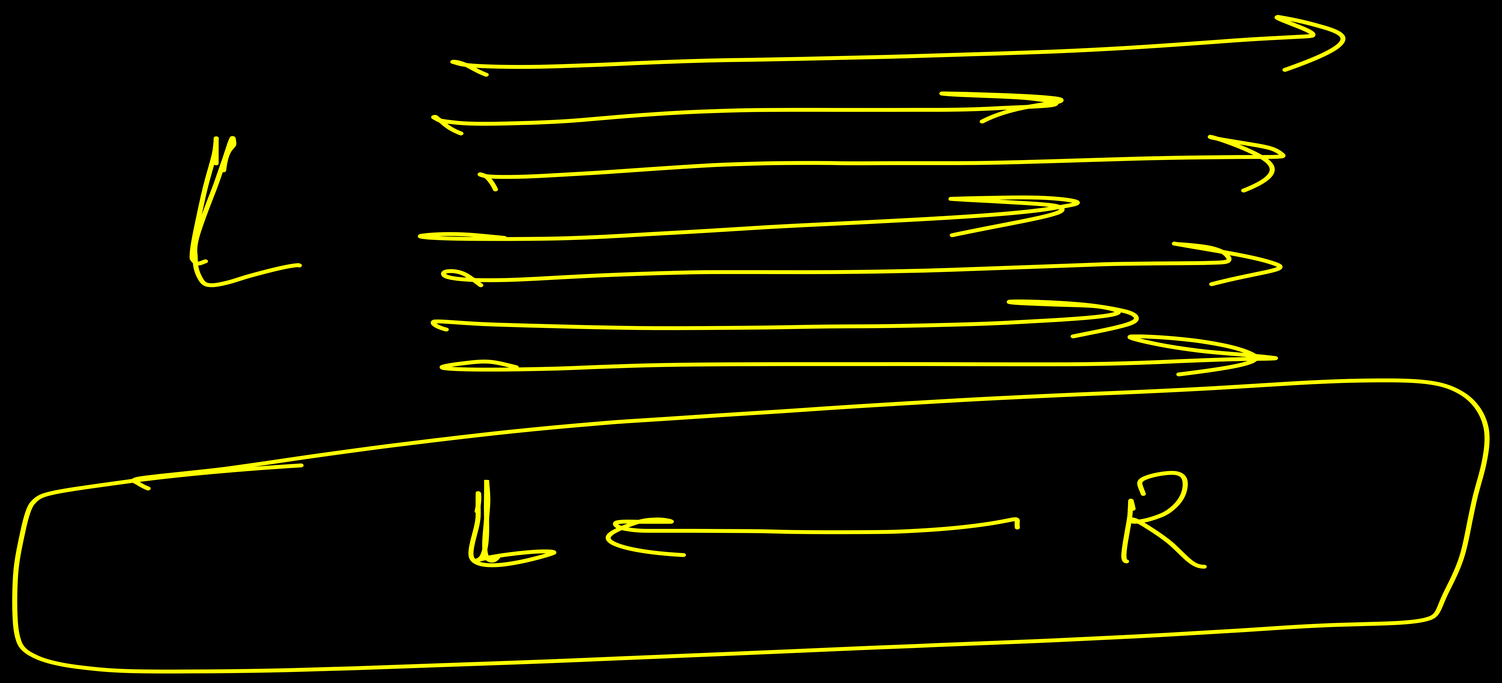
- 5 is greater than all the elements to its right [2], so it is a leader.
- 2 has no elements to its right, so it is a leader.

Appr 2 : Optimised :

[an elem greater than or equal to all the elements on its right]

⇓

[an elem greater than or equal to the max elem on its right.]



max_elem so far.

(curr - elem ? = max elem)

max_elem = curr & Locate

[NAT]

#Q. Missing and Repeating in an Array

Problem Statement : Given an unsorted array of size n, where the array elements are in the range 1 to n, one number from the set {1, 2, ..., n} is missing, and one number occurs twice. The task is to find these two numbers. Examples

Example 1:

Input: arr[] = {3, 1, 3}

Output: 3, 2

Explanation:

- 3 occurs twice in the array.
- 2 is missing from the array.

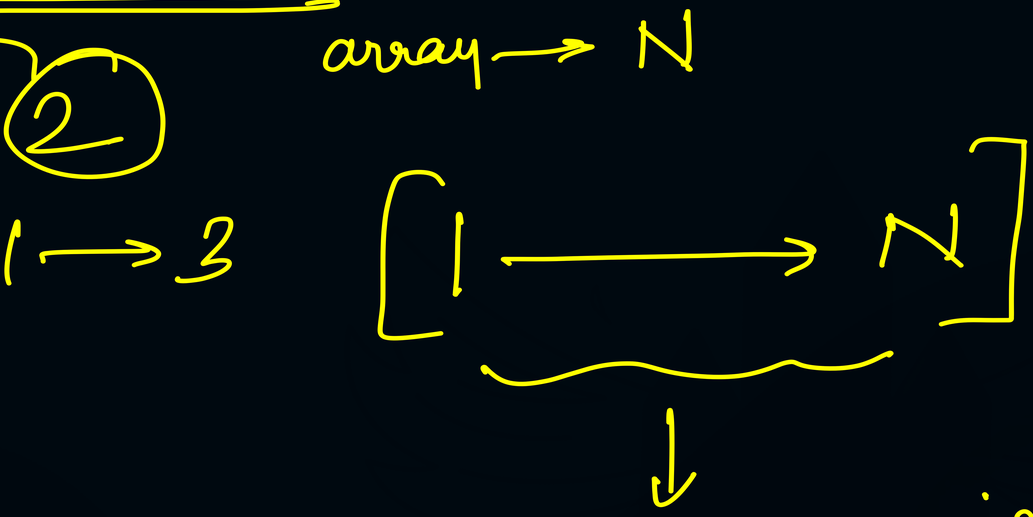
Example 2:

Input: arr[] = {4, 3, 6, 2, 1, 1}

Output: 1, 5

Explanation:

- 1 occurs twice in the array.
- 5 is missing from the array.



Appl :-> Sort

1 3 3

```
if (a[i] != i+1)
{
    missing_elem = i+1 ;
}
if (a[i] == a[i+1])
{
    duplicate = a[i]
}
```

[NAT]

#Q. Stock Buy and Sell – Max one Transaction Allowed

Problem Statement: Given an array prices[] of length N, representing the prices of stocks on different days, the task is to find the maximum profit possible by buying and selling the stocks on different days when at most one transaction is allowed.

Here, one transaction means one buy and one sell.

Note: The stock must be bought before being sold.

Examples

Example 1:

Input: prices[] = {7, 10, 1, 3, 6, 9, 2}

Output: 8

Explanation:

- Buy at 1 and sell at 9 to get a profit of 9 - 1 = 8.

Example 2:

Input: prices[] = {7, 6, 4, 3, 1}

Output: 0

1B 1S

max min

10 - 1 = 9 X

1B 1S

B / S

Special Case

at most 1 txn → ≤ 1

0, 1

8, 7, 3, 1

loss

buy & sell

buy - sell

≥ 0

$m_1 =$ max (array)

$m_2 =$ min (array)

ans → $m_1 - m_2$?

Appr1 :-

profit = 0

for i : 0 ⟶ n-1 buying on ith day

j : i+1 ⟶ n selling on jth day.

profit = max (profit, a(j) - a(i))

Optimised Soln :-

min-so-far

buy

curr - date
selling
date

min on left

sell

buy date

min_elem = a(0).

profit = 0

For (i = 1 ⟶ n)

{ min_elem = min (min_elem , a(i));

profit = max (profit , a(i) - min_elem)

}

Explanation:

- Since the array is sorted in decreasing order, no profit can be made.

Example 3:

Input: prices[] = {1, 3, 6, 9, 11}

Output: 10

Explanation:

- The array is sorted in increasing order, so we can make the maximum profit by buying at 1 and selling at 11, yielding a profit of 11 - 1 = 10.

[MCQ]

H-W

very famous

#Q. Stock Buy and Sell – Multiple Transaction Allowed

Problem Statement: Given an array prices[] of size n denoting the cost of stock on each day, the task is to find the maximum total profit if we can buy and sell the stocks any number of times.

Constraints:

- We can only sell a stock that we have previously bought.
- We cannot hold multiple stocks at the same time (i.e., we must sell before buying again).

Imp

Examples

Example 1:

Input: prices[] = {100, 180, 260, 310, 40, 535, 695}

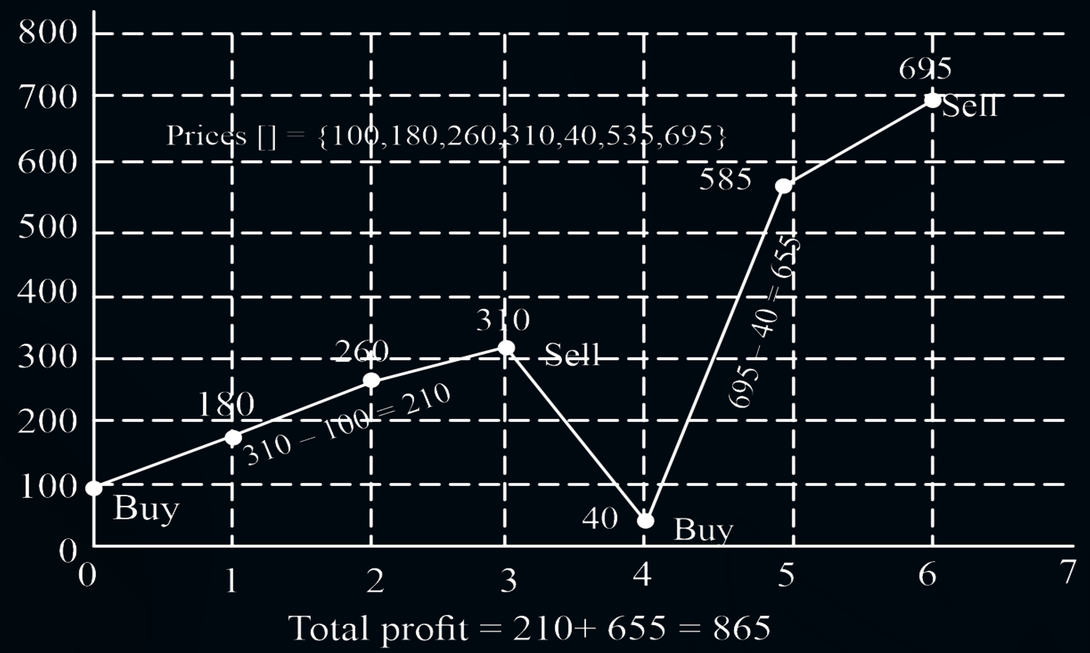
Output: 865

Explanation:

- Buy on Day 0 and Sell on Day 3 → 310 - 100 = 210
- Buy on Day 4 and Sell on Day 6 → 695 - 40 = 655
- Total Maximum Profit = 210 + 655 = 865

Txn 1

Txn 2



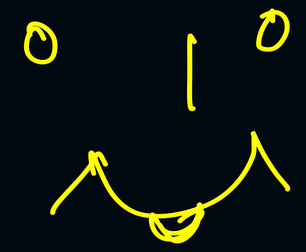
Example 2:

Input: prices[] = {4, 2, 2, 2, 4}

Output: 2

Explanation:

- Buy on Day 3 and Sell on Day 4 → 4 - 2 = 2
- Total Maximum Profit = 2



THANK - YOU