TCS | NQT

NATIONAL QUALIFIER TEST

ARRAYS & STRINGS-1

CS

About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored working professions in field of Data Science and Analytics 10K+
11. Have been mentoring GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

Telegram

Telegram Link for Aditya Jain sir:
https://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

[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]

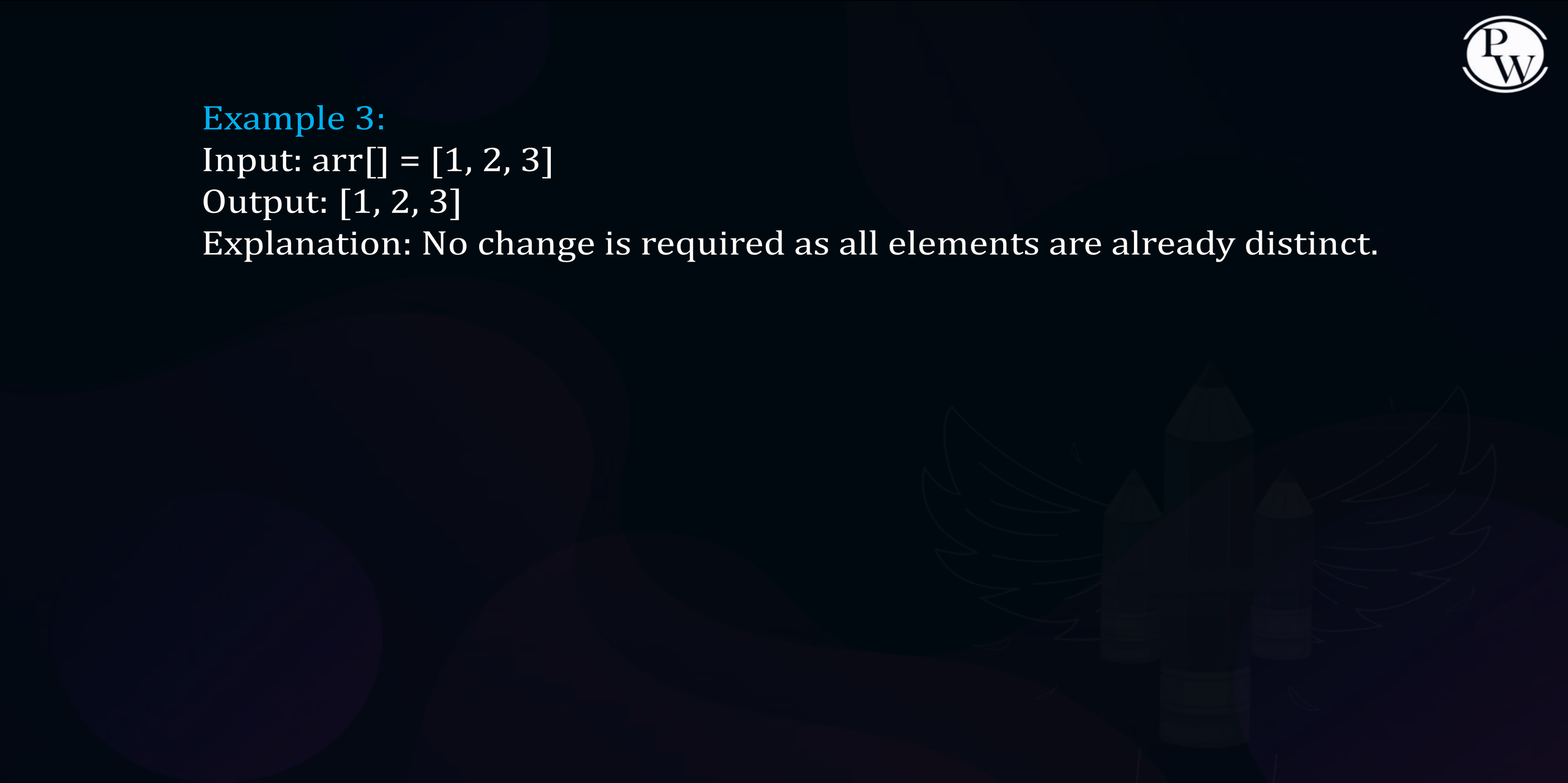
Explanation: All distinct elements are extracted and placed at the beginning in sorted order.

Example 3:

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

Output: [1, 2, 3]

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



[MCQ]

#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.

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.

[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:

[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

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]

#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).

Examples

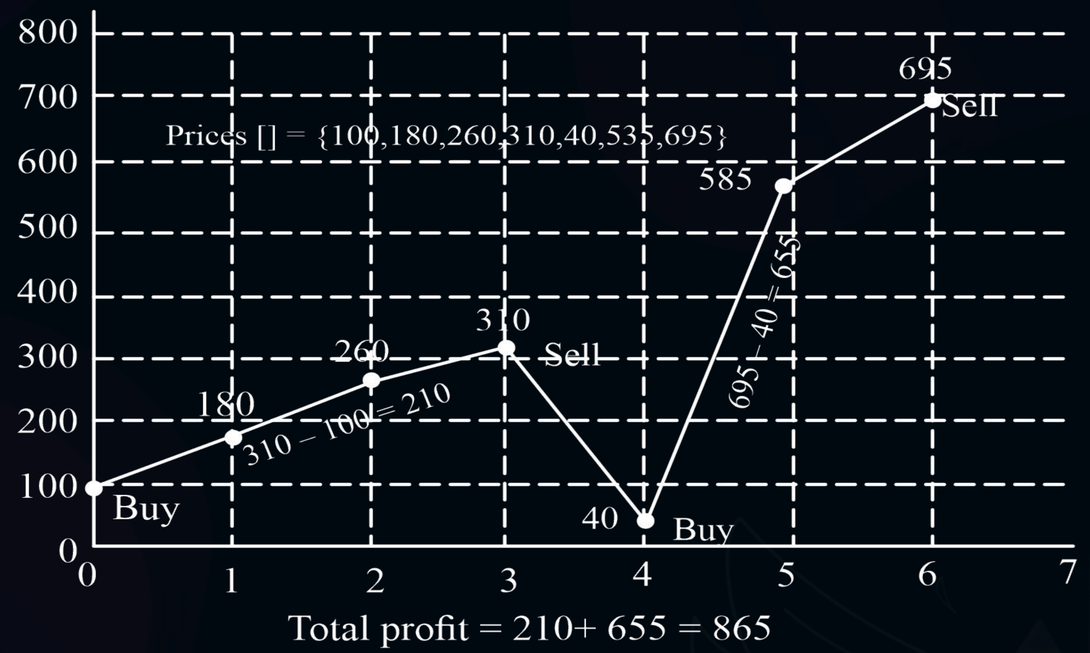
Example 1:

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



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

[MCQ]

#Q. Product of Array Except Self

Problem Statement: Given an array arr[] of n integers, construct a product array res[] (of the same size) such that res[i] is equal to the product of all elements of arr[] except arr[i].

Examples

Example 1:

Input: arr[] = [10, 3, 5, 6, 2]

Output: [180, 600, 360, 300, 900]

Explanation:

- res[0] = 3 * 5 * 6 * 2 = 180
- res[1] = 10 * 5 * 6 * 2 = 600
- res[2] = 10 * 3 * 6 * 2 = 360
- res[3] = 10 * 3 * 5 * 2 = 300
- res[4] = 10 * 3 * 5 * 6 = 900

Example 2:

Input: arr[] = [12, 0]

Output: [0, 12]

Explanation:

- res[0] = 0 (since any number multiplied by 0 is 0).
- res[1] = 12 (since all other elements except arr[1] are multiplied).

[NAT]

#Q. Maximum Subarray Sum – Kadane's Algorithm

Problem Statement: Given an array arr[], the task is to find the subarray that has the maximum sum and return its sum.

Examples

Example 1:

Input: arr[] = {2, 3, -8, 7, -1, 2, 3}

Output: 11

Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.

Example 2:

Input: arr[] = {-2, -4}

Output: -2

Explanation: The subarray {-2} has the largest sum -2.

Example 3:

Input: arr[] = {5, 4, 1, 7, 8}

Output: 25

Explanation: The subarray {5, 4, 1, 7, 8} has the largest sum 25.

[NAT]

#Q. Find Smallest Missing Positive Number

Problem Statement: Given an unsorted array arr[] containing both positive and negative elements, the task is to find the smallest positive number missing from the array.

Note: You can modify the original array.

Examples

Example 1:

Input: arr[] = {2, -3, 4, 1, 1, 7}

Output: 3

Explanation: The positive numbers in sorted order are {1, 2, 4, 7}. The smallest missing positive number is 3.

Example 2:

Input: arr[] = {5, 3, 2, 5, 1}

Output: 4

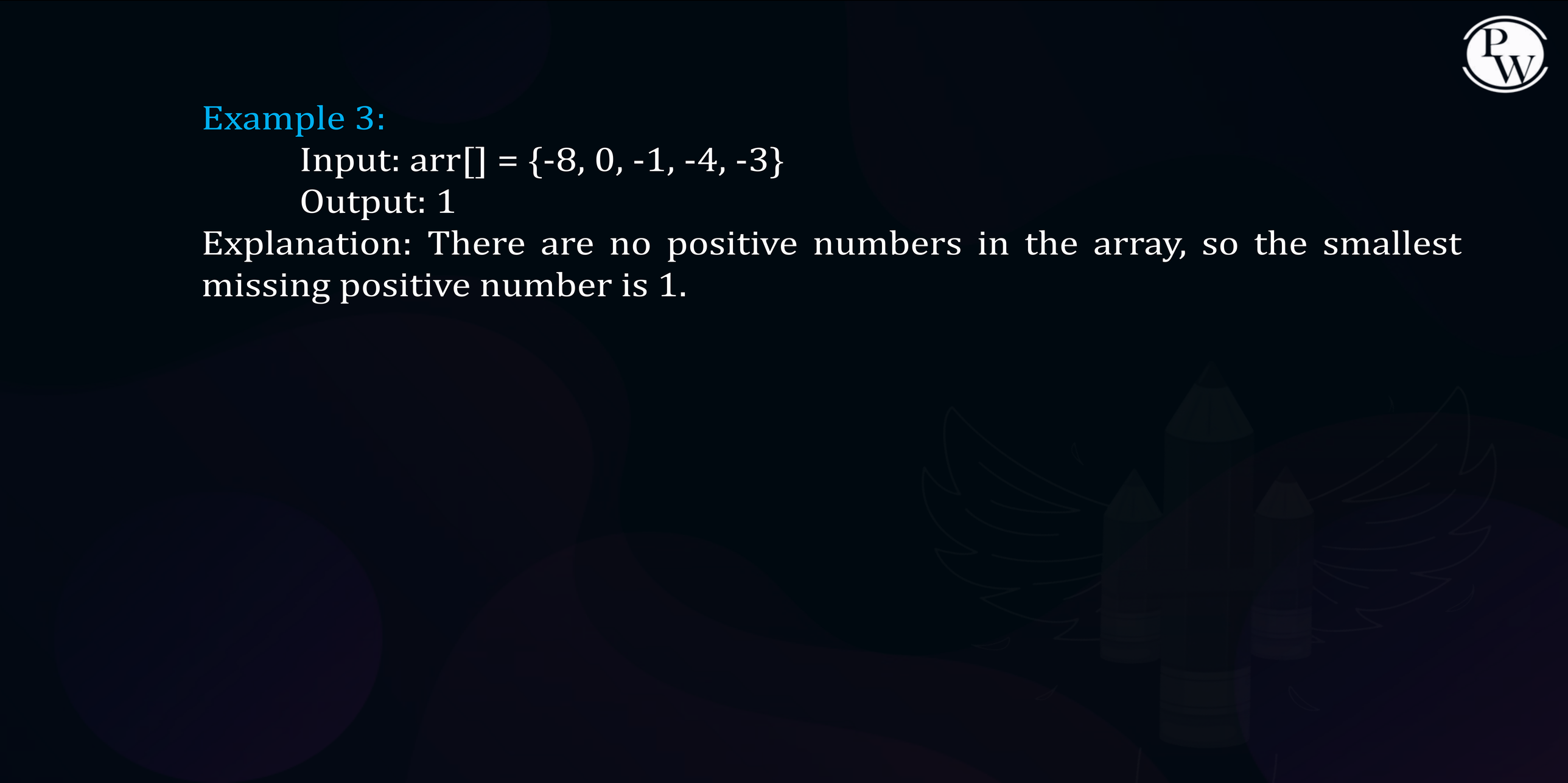
Explanation: The positive numbers in sorted order are {1, 2, 3, 5, 5}. The smallest missing positive number is 4.

Example 3:

Input: arr[] = {-8, 0, -1, -4, -3}

Output: 1

Explanation: There are no positive numbers in the array, so the smallest missing positive number is 1.



[NAT]

#Q. Third largest element in an array of distinct elements

Problem Statement: Given an array of n distinct integers, the task is to find the third largest element in the array.

Examples

Example 1:

Input: arr[] = {1, 14, 2, 16, 10, 20}

Output: 14

Explanation:

- Largest element: 20
- Second largest element: 16
- Third largest element: 14

Example 2:

Input: arr[] = {19, -10, 20, 14, 2, 16, 10}

Output: 16

Explanation:

[MCQ]

#Q. Sort an array in wave form

Problem Statement: Given an unsorted array of integers, sort the array into a wave array.

An array arr[0..n-1] is in wave form if:

- arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4] >= ...

In other words, the elements should alternate between high and low values in a wave-like pattern.

Examples

Example 1:

Input: arr[] = {10, 5, 6, 3, 2, 20, 100, 80}

Output: arr[] = {10, 5, 6, 2, 20, 3, 100, 80}

Aditya Jain Sir PW

Explanation:

- The elements follow a wave pattern: large - small - large - small - large - small
- Other valid wave forms could be {20, 10, 6, 2, 100, 3, 80, 5}

Example 2:

Input: arr[] = {20, 10, 8, 6, 4, 2}

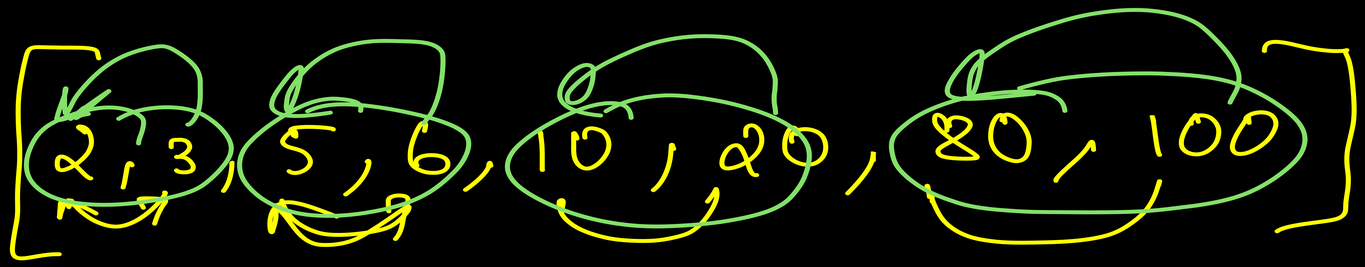
Output: arr[] = {20, 8, 10, 4, 6, 2}

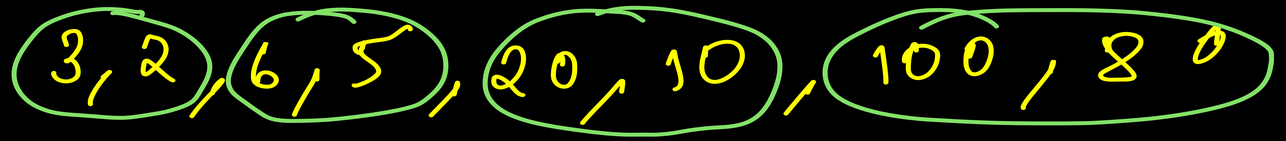
Explanation:

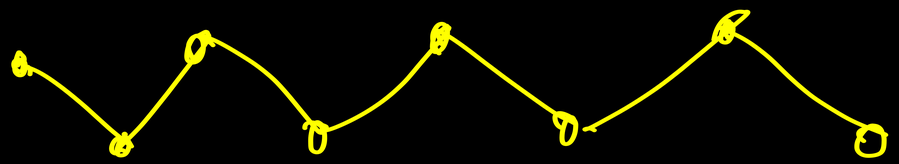
- The wave pattern is maintained: large - small - large - small - large - small
- Other valid wave forms could be {10, 20, 6, 8, 2, 4}

10 , 5 , 6 , 3 , 2 , 20 , 100 , 80

||



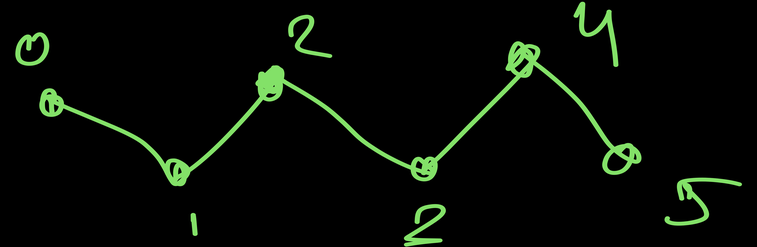




Sorting appr ⟶ O(nlogn)

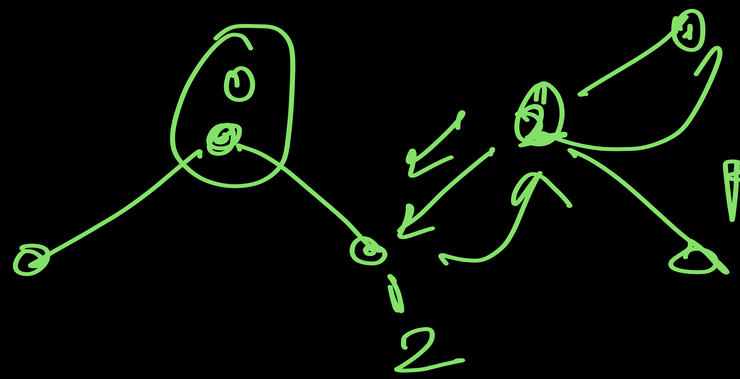
Appr 2 :-

Array → even index elements

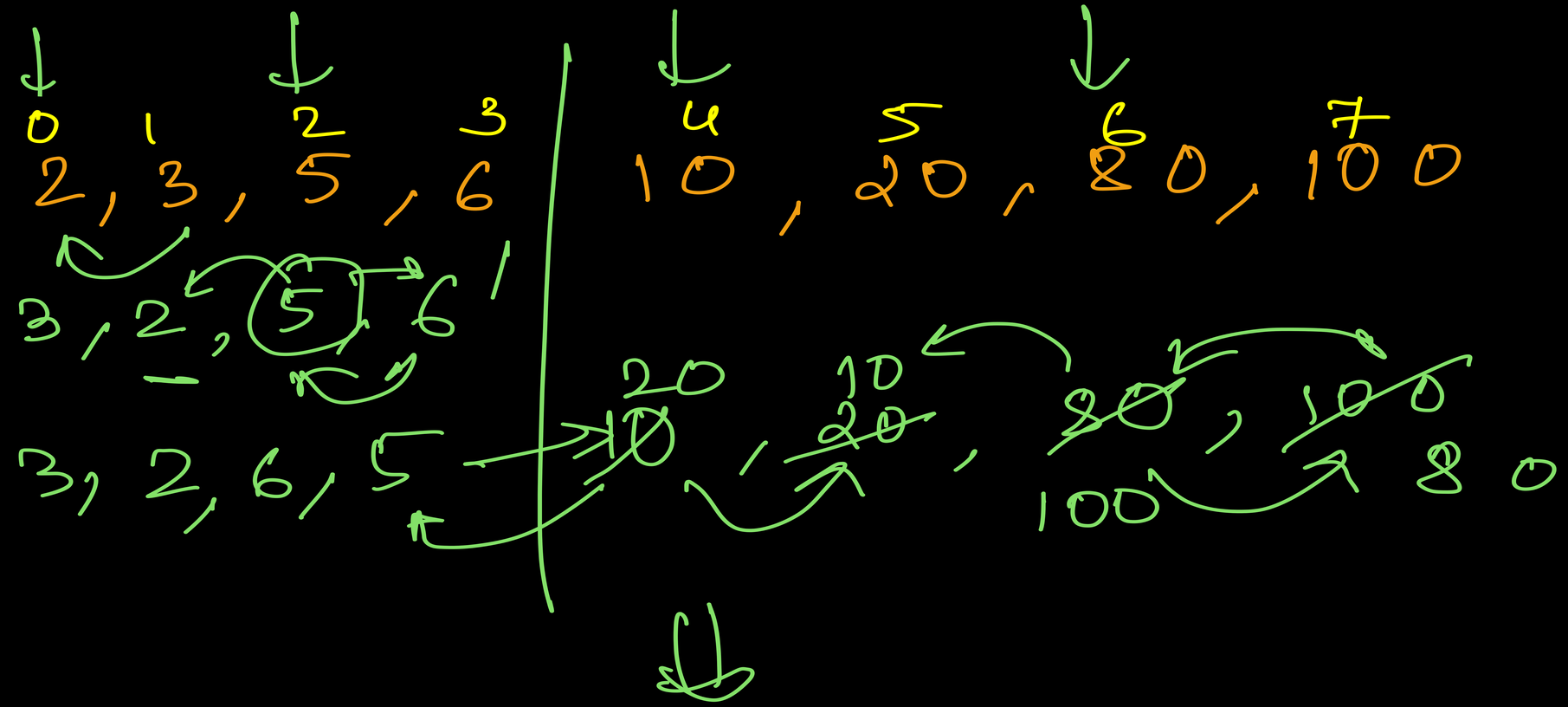


prev-odd

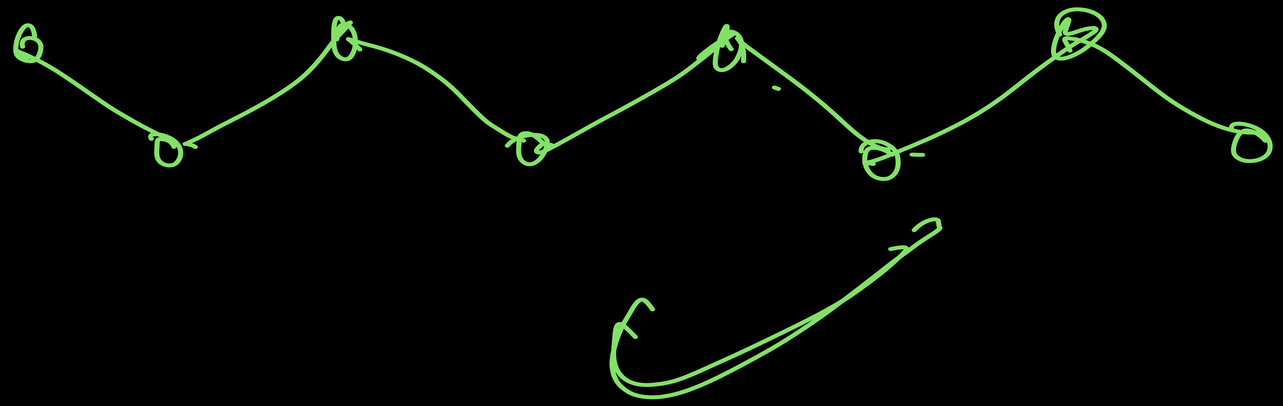
curr → swap ⇒ curr, prev_odd



prev-odd, next-odd · curr, next odd, prev



3, 2, 6, 5, 20, 10, 100, 80



[NAT]

#Q. Split array into three equal sum segments

Problem statement : Given an integer array arr[], the task is to divide the array into three non-empty contiguous segments with equal sum. In other words, we need to return an index pair [i, j], such that sum(arr[0…i]) = sum(arr[i+1…j]) = sum(arr[j+1…n-1]).

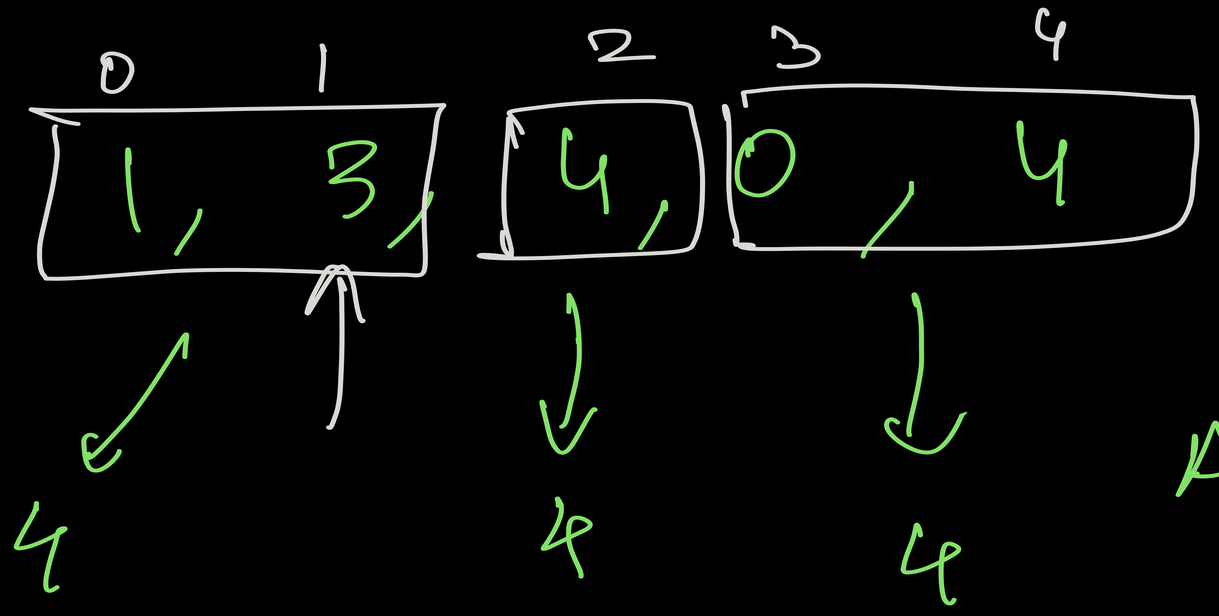
Note: If it is impossible to divide the array into three non-empty contiguous segments, return [-1, -1].

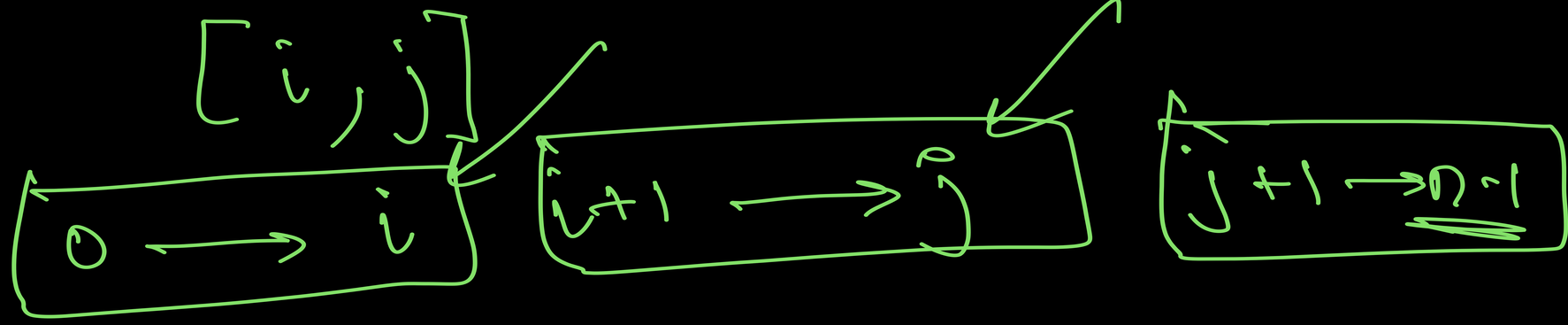
"Aditya Jain Sir PW"

Examples:

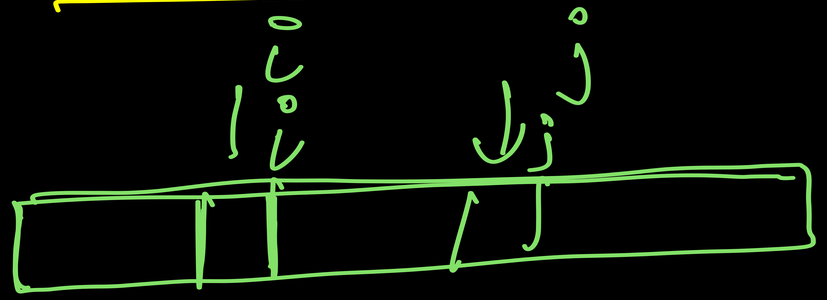
- Input: arr[] = [1, 3, 4, 0, 4] Output: [1, 2] Explanation: 3 equal sum segments are: arr[0…1], arr[2…2] and arr[3…4] each having sum = 4.
- Input: arr[] = [2, 3, 4] Output: [-1, -1] Explanation: No three segments exist which has equal sum.
- Input: arr[] = [1, -1, 1, -1, 1, -1, 1, -1] Output: [1, 3] Explanation: 3 equal sum segments are: arr[0…1], arr[2…3] and arr[4…7] each having sum = 0.

2 | 3 | 4


0 1 2 3 4
1, 3, 4, 0, 4
4
4
4
Ans: [1, 2]


[i, j]
0 → i
i+1 → j
j+1 → n-1

Soln :- Appr1 : Brute Force :

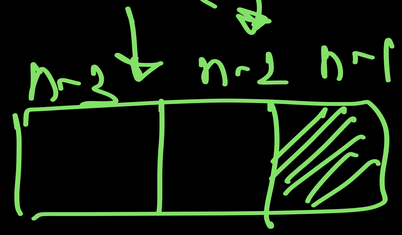


find (i, j) such that :

$[a_0 \rightarrow a_i]$ $[a_{i+1} \rightarrow a_j]$ $(a_{j+1} \rightarrow a_{n-1})$

↳ $i = 0$ & $i < n-2$

↳ $j = i+1$; $j < n-1$



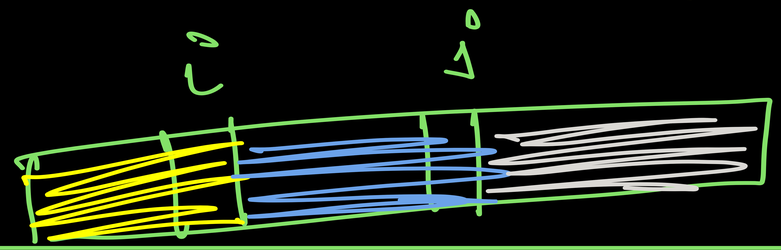
for (i = 0 ⟶ < n - 2)

for (j = i + 1 ⟶ < n - 1)

$$\begin{cases} sum1 = findSum(a, 0, i) \\ sum2 = findSum(a, i+1, j) \\ sum3 = findSum(a, j+1, n-1) \end{cases}$$

(sum1 == sum2) == sum3)

Above code Complexity : $O(n^3)$




```
For(0 ⟶ n-1)
    t_sum += a(i)
```

⟶ if (t_sum % 3 != 0)
    return {-1, -1}

sum = 0 , int c = 0

for (i = 0 ⟶ n - 1)

{ sum += a[i]

if (sum == t - sum/3) 'i'

{ aus[c] = i

sum = 0

c++

if (c == 2)

break

}

if (c == 2)

found → ans(0), ans(1)

else

not found {-1, -1}

(Q) Given an array in non-decreasing order. find the index of the first occurance of the given elem.

eg:- A : [2, 3, 5, 5, 5, 5, 6, 6, 7]

0 1 2 3 4 5 6 7 8

x = 5

// Ans :- 2

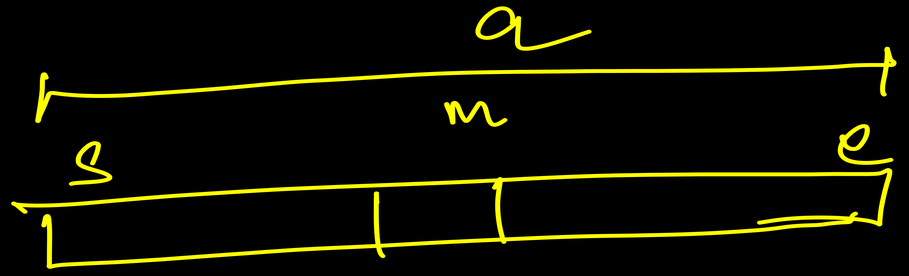
Appr :-

```
ans = -1;
for (i = 0 ⟶ n-1)
{
    if (A(i) == x)
    {
        ans = i
        break;
    }
}
```

⟶ O(n)

Better Soln :-

Binary Search based



$m = (s+e)/2$

while (left <= right)

if (a(m) == x)

↳ stop

else if (x < a(m))

↳ explore left $\quad e = m-1$

x > a(m) ↳ else ⟶ explore right $\quad s = m+1$

THANK - YOU