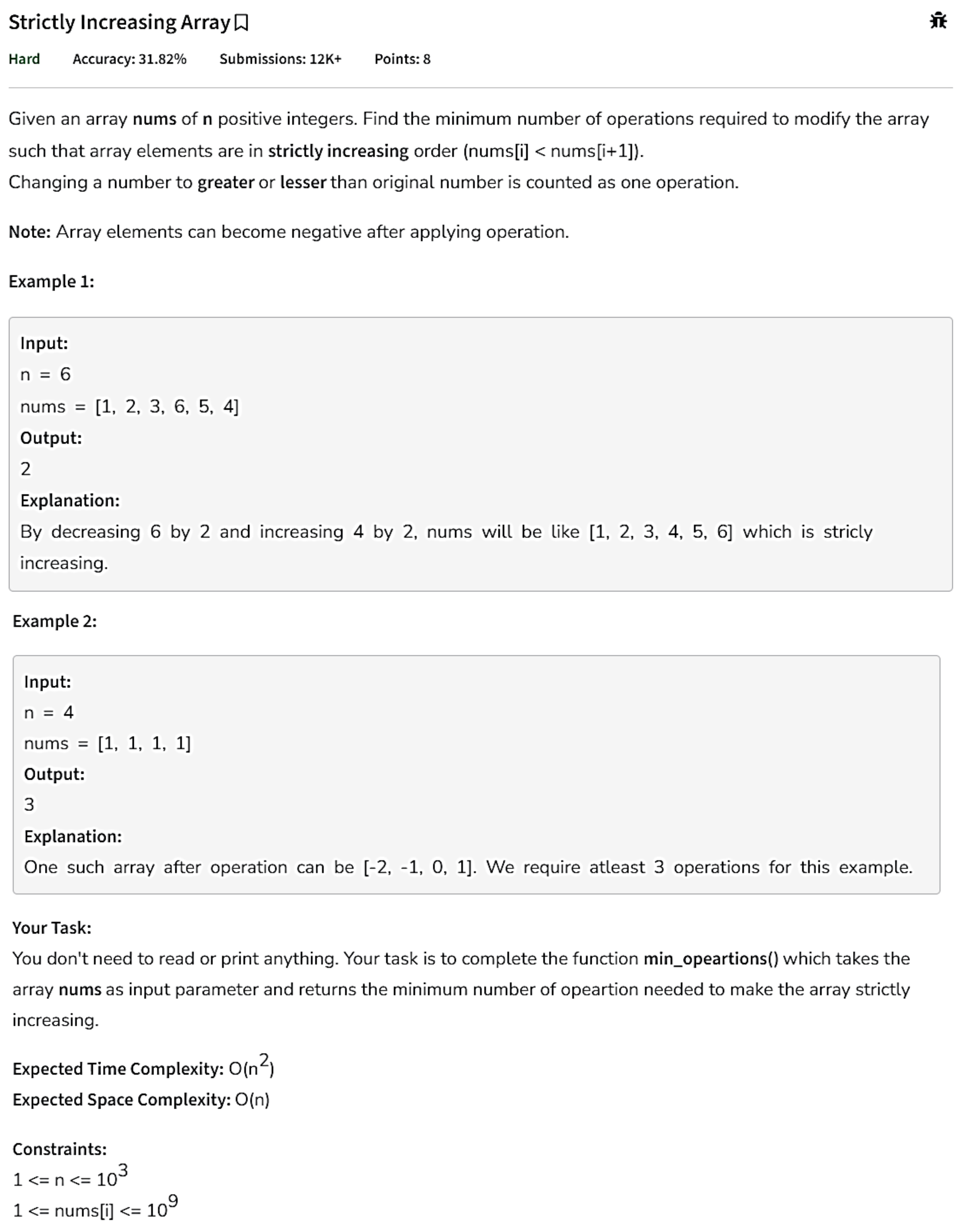
PROBLEM



Strictly Increasing Array

Hard **Accuracy: 31.82%** **Submissions: 12K+** **Points: 8**

Given an array **nums** of **n** positive integers. Find the minimum number of operations required to modify the array such that array elements are in **strictly increasing** order (nums[i] < nums[i+1]).
Changing a number to **greater** or **lesser** than original number is counted as one operation.

Note: Array elements can become negative after applying operation.

Example 1:

```
Input:
n = 6
nums = [1, 2, 3, 6, 5, 4]
Output:
2
Explanation:
By decreasing 6 by 2 and increasing 4 by 2, nums will be like [1, 2, 3, 4, 5, 6] which is stricly
increasing.
```

Example 2:

```
Input:
n = 4
nums = [1, 1, 1, 1]
Output:
3
Explanation:
One such array after operation can be [-2, -1, 0, 1]. We require atleast 3 operations for this example.
```

Your Task:
You don't need to read or print anything. Your task is to complete the function **min_opeartions()** which takes the array **nums** as input parameter and returns the minimum number of opeartion needed to make the array strictly increasing.

Expected Time Complexity: $O(n^2)$
Expected Space Complexity: $O(n)$

Constraints:
$1 <= n <= 10^3$
$1 <= nums[i] <= 10^9$

CODE

```
#User function Template for python3


class Solution:
    # Function to calculate the Longest Increasing Subsequence (LIS)

    def LIS(self,arr,n):

        # Initializing a result list with all elements set to 1

        res = [1]*n


        # Nested loops to compare all pairs of elements in the array

        for i in range(1,n):

            for j in range(i):

                # Checking if the current element is greater than the previous element

                # and if the difference between the current and previous element is equal to the
difference

                # in their indices. If this condition is satisfied, update the value in the result list.

                if arr[j] < arr[i] and arr[i]-arr[j] >=  i-j:

                    res[i] = max(res[i],res[j] + 1)


        # Return the result list which represents the length of the longest increasing
subsequence for each element.

        return res


    # Function to find the minimum number of operations required

    def min_operations(self,nums):

        # Get the length of the input list

        n = len(nums)


        # Calculate the Longest Increasing Subsequence

        lis = self.LIS(nums,n)


   # Return the difference between the input list length and the maximum value in the LIS list

        return n-max(lis)
```