**Question 1 : Leader in the Array Given a unsorted array, kindly find the leader in array . An element is called the leader of an array if there is no element greater than it on the right side.**

**Test case**

**int arr[] = {7, 10, 4, 10, 6, 5, 2}, n = 7; 10 6 5 2**

**Constraint Please submit the optimize approach in 0(n)**

**def sort\_leaders(arr):**

**n = len(arr)**

**maximum\_right = arr[n - 1]**

**leaders = [maximum\_right]**

**for i in range(n - 2, -1, -1):**

**if arr[i] >= maximum\_right:**

**leaders.append(arr[i])**

**maximum\_right = arr[i]**

**return leaders[::-1]**

**Test case**

**arr = [7, 10, 4, 10, 6, 5, 2]**

**leaders = sort\_leaders(arr)**

**print(“Leaders ", leaders)**

**Output-**

**Leaders in the array: [10, 6, 5, 2]**

**Question 2 :**

**Best Time to Buy and Sell Stock You are given an array prices where prices[i] is the price of a given stock on the ith day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock. Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0. Input: prices = [7,1,5,3,6,4] Output: 5 Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5. Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell Input: prices = [7,6,4,3,1] Output: 0 Explanation: In this case, no transactions are done and the max profit = 0. Constraints: 1 <= prices.length <= 105 0 <= prices[i] <= 104**

**def maximum\_profit(prices):**

**if not prices:**

**return 0**

**minimum\_price = prices[0]**

**maximum\_profit = 0**

**for price in prices:**

**minimum\_price = min(minimum\_price, price)**

**maximum\_profit = max(maximum\_profit, price - minimum\_price)**

**return maximum\_profit**

**Test cases**

**prices1 = [7, 1, 5, 3, 6, 4]**

**prices2 = [7, 6, 4, 3, 1]**

**print("Maximum profit ", maximum\_profit (prices1)) Output: 5**

**print("Maximum profit ", maximum\_profit (prices2)) Output: 0**

**Question 3:**

**Sum of All Subset XOR Totals The XOR total of an array is defined as the bitwise XOR of all its elements, or 0 if the array is empty. For example, the XOR total of the array [2,5,6] is 2 XOR 5 XOR 6 = 1. Given an array nums, return the sum of all XOR totals for every subset of nums. Note: Subsets with the same elements should be counted multiple times. An array a is a subset of an array b if a can be obtained from b by deleting some (possibly zero) elements of b. Input: nums = [1,3] Output: 6 Explanation: The 4 subsets of [1,3] are: - The empty subset has an XOR total of 0. - [1] has an XOR total of 1. - [3] has an XOR total of 3. - [1,3] has an XOR total of 1 XOR 3 = 2. 0 + 1 + 3 + 2 = 6 Input: nums = [5,1,6] Output: 28 Explanation: The 8 subsets of [5,1,6] are: - The empty subset has an XOR total of 0. - [5] has an XOR total of 5. - [1] has an XOR total of 1. Softnerve Tech Assement - [6] has an XOR total of 6. - [5,1] has an XOR total of 5 XOR 1 = 4. - [5,6] has an XOR total of 5 XOR 6 = 3. - [1,6] has an XOR total of 1 XOR 6 = 7.**

**def subsetXORSum(number):**

**def backtrack(index, current\_xor):**

**nonlocal total\_xor**

**total\_xor += current\_xor**

**for i in range(index, len(number)):**

**backtrack(i + 1, current\_xor ^ number[i])**

**total\_xor = 0**

**backtrack(0, 0)**

**return total\_xor**

**Test cases**

**number1 = [1, 3]**

**number2 = [5, 1, 6]**

**print("Output :", subsetXORSum(number1)) # Output: 6**

**print("Output ", subsetXORSum(number2)) # Output: 28**