**Experiment 4**

|  |  |
| --- | --- |
| **Student Name:** Dhruv Sumbria | **UID:**22BET10262 |
| **Branch:** BE -IT | **Section/Group:**22BET/IOT/702/B |
| **Semester:** 6th | **Date of Performance:**14/02/2025 |
| **Subject Name:** Advanced Programming Lab-2 | **Subject Code:** 22ITP-351 |

1. **Aim 1: Longest Nice Substring**

A string s is nice if, for every letter of the alphabet that s contains, it appears both in uppercase and lowercase. For example, "abABB" is nice because 'A' and 'a' appear, and 'B' and 'b' appear. However, "abA" is not because 'b' appears, but 'B' does not.

### **Reverse Bits :**

Note that in some languages, such as Java, there is no unsigned integer type. In this case, both input and output will be given as a signed integer type. They should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned.

1. [**Number of 1 Bits**](https://leetcode.com/problems/number-of-1-bits/) **:**

Given a positive integer n, write a function that returns the number ofset bitsin its binary representation (also known as the [Hamming weight](http://en.wikipedia.org/wiki/Hamming_weight)).

1. [**Maximum Subarray**](https://leetcode.com/problems/maximum-subarray/) :

Given an integer array nums, find the subarraywith the largest sum, and return its sum.

1. [**Search a 2D Matrix II**](https://leetcode.com/problems/search-a-2d-matrix-ii/) :

Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. This matrix has the following properties:

* Integers in each row are sorted in ascending from left to right.
* Integers in each column are sorted in ascending from top to bottom.

1. [**Super Pow**](https://leetcode.com/problems/super-pow/) **:**

Your task is to calculate ab mod 1337 where a is a positive integer and b is an extremely large positive integer given in the form of an array.

1. [**Beautiful Array**](https://leetcode.com/problems/beautiful-array/) :

An array nums of length n is beautiful if:

* nums is a permutation of the integers in the range [1, n].
* For every 0 <= i < j < n, there is no index k with i < k < j where 2 \* nums[k] == nums[i] + nums[j].

Given the integer n, return any beautiful array nums of length n. There will be at least one valid answer for the given n.

1. [**The Skyline Problem**](https://leetcode.com/problems/the-skyline-problem/) :

A city's **skyline** is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return the **skyline** formed by these buildings collectively.

The geometric information of each building is given in the array buildings where buildings[i] = [lefti, righti, heighti]:

* lefti is the x coordinate of the left edge of the ith building.
* righti is the x coordinate of the right edge of the ith building.
* heighti is the height of the ith building.

1. [**Reverse Pairs**](https://leetcode.com/problems/reverse-pairs/) :

Given an integer array nums, return the number of **reverse pairs** in the array.

A **reverse pair** is a pair (i, j) where:

* 0 <= i < j < nums.length and
* nums[i] > 2 \* nums[j].

1. [**Longest Increasing Subsequence II**](https://leetcode.com/problems/longest-increasing-subsequence-ii/) :

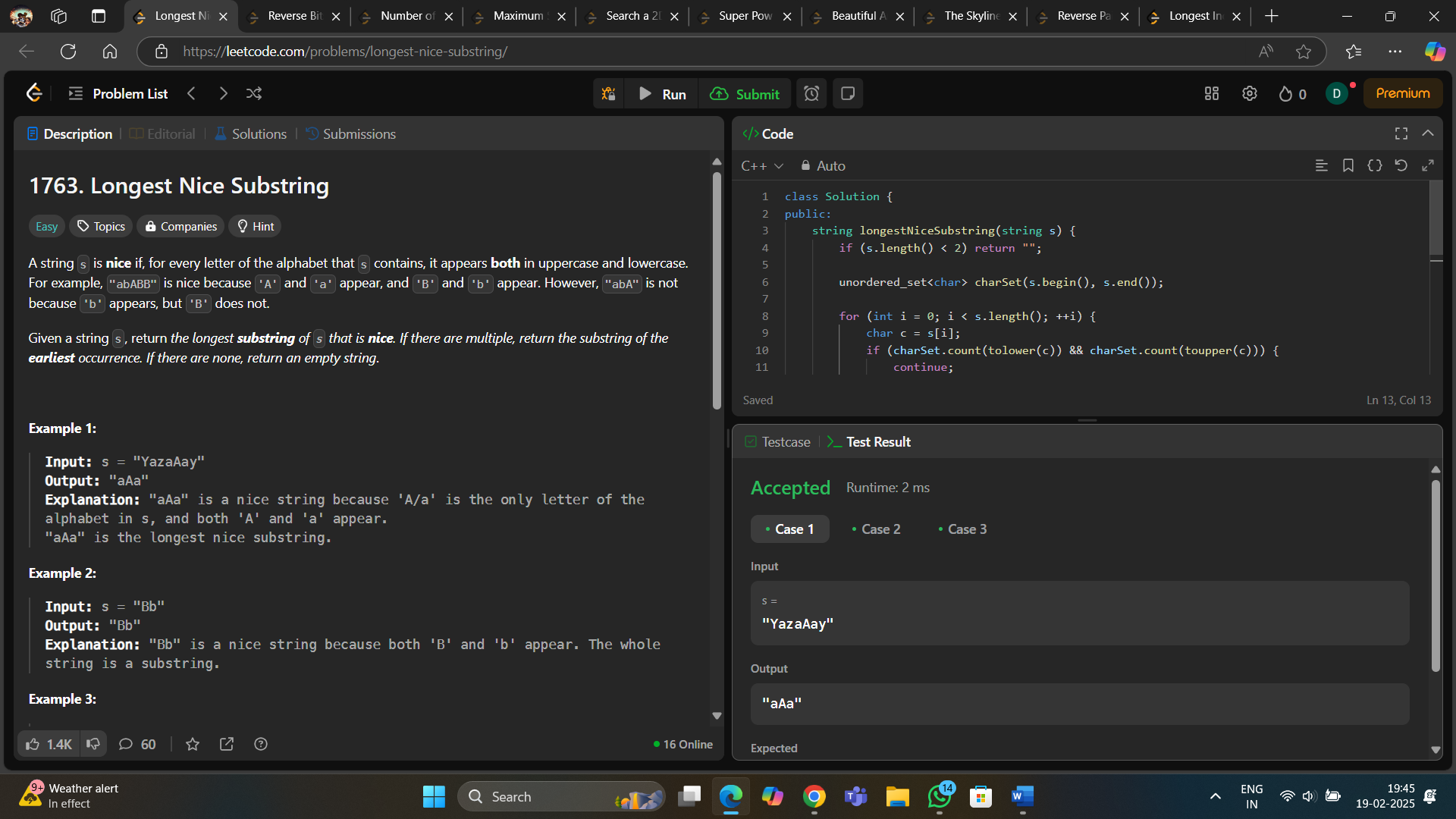
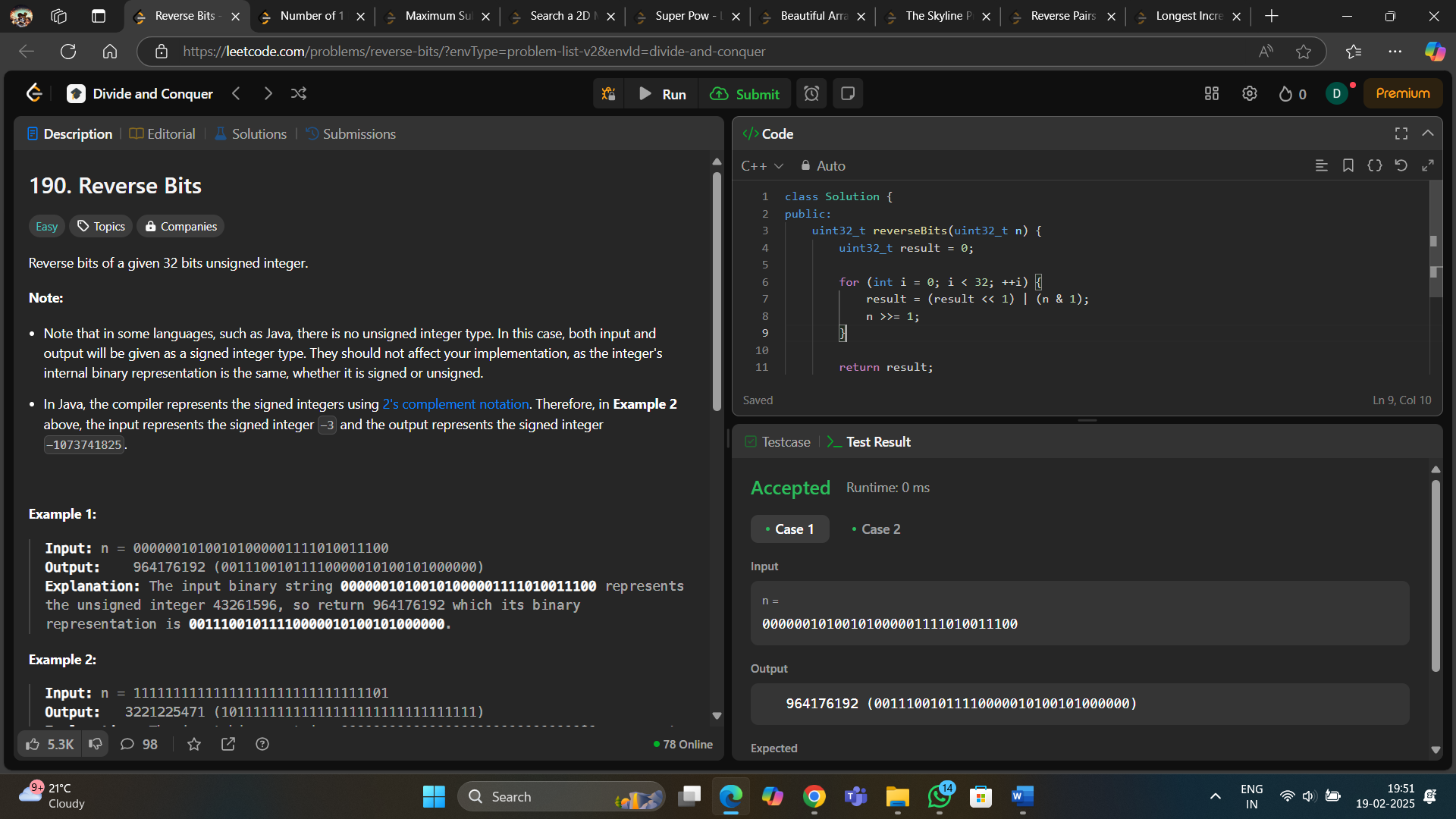
You are given an integer array nums and an integer k.

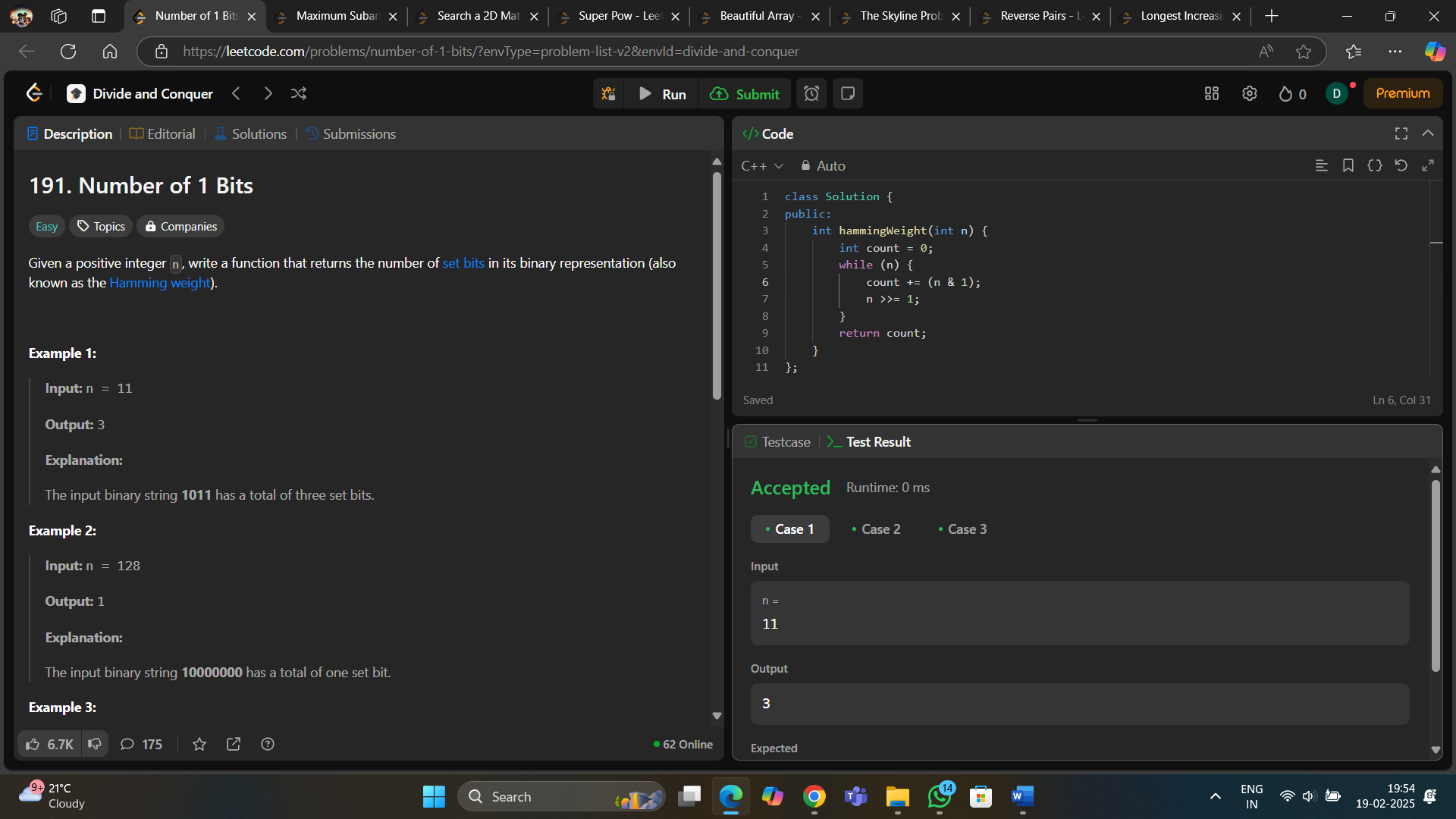
Find the longest subsequence of nums that meets the following requirements:

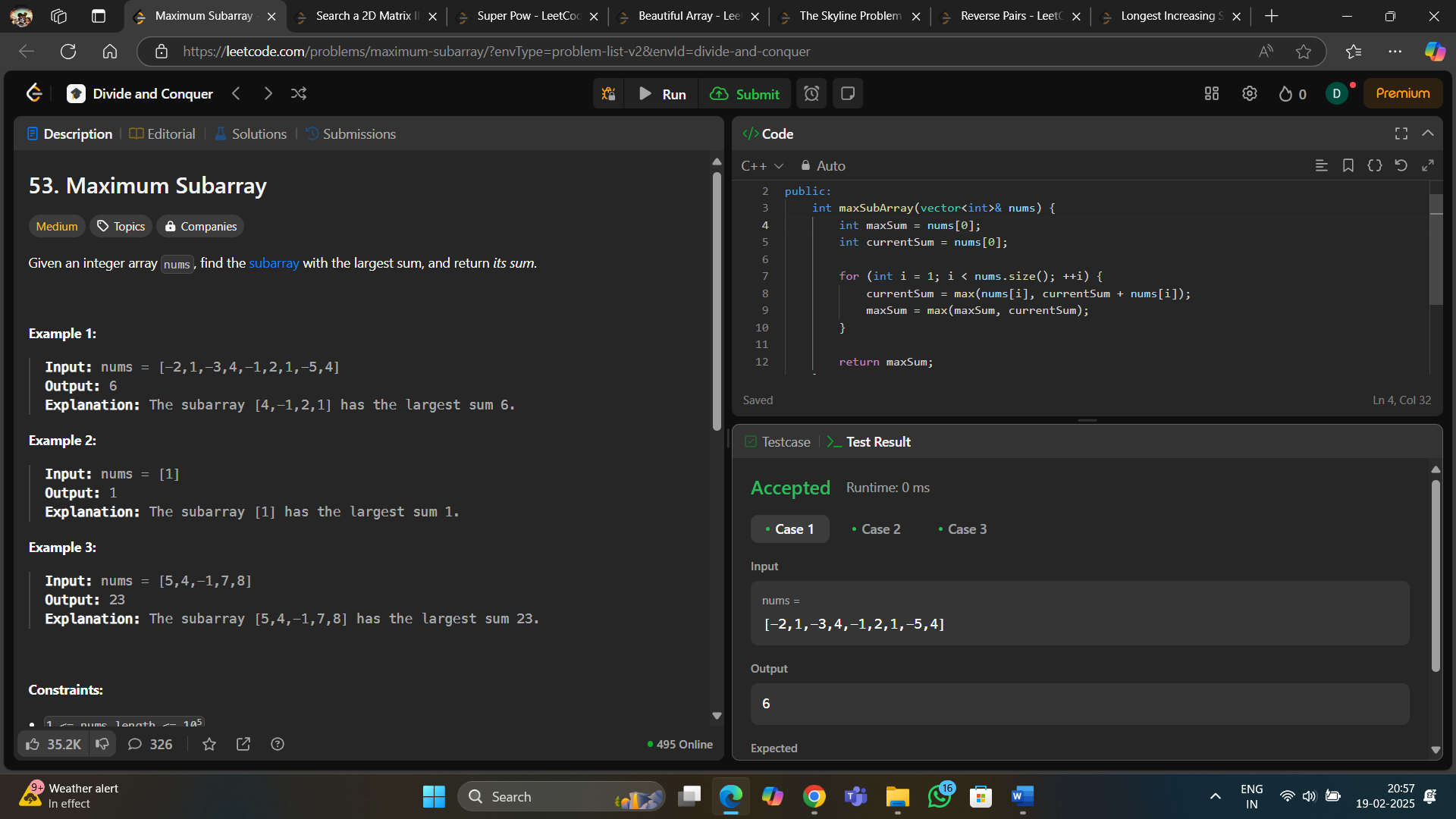
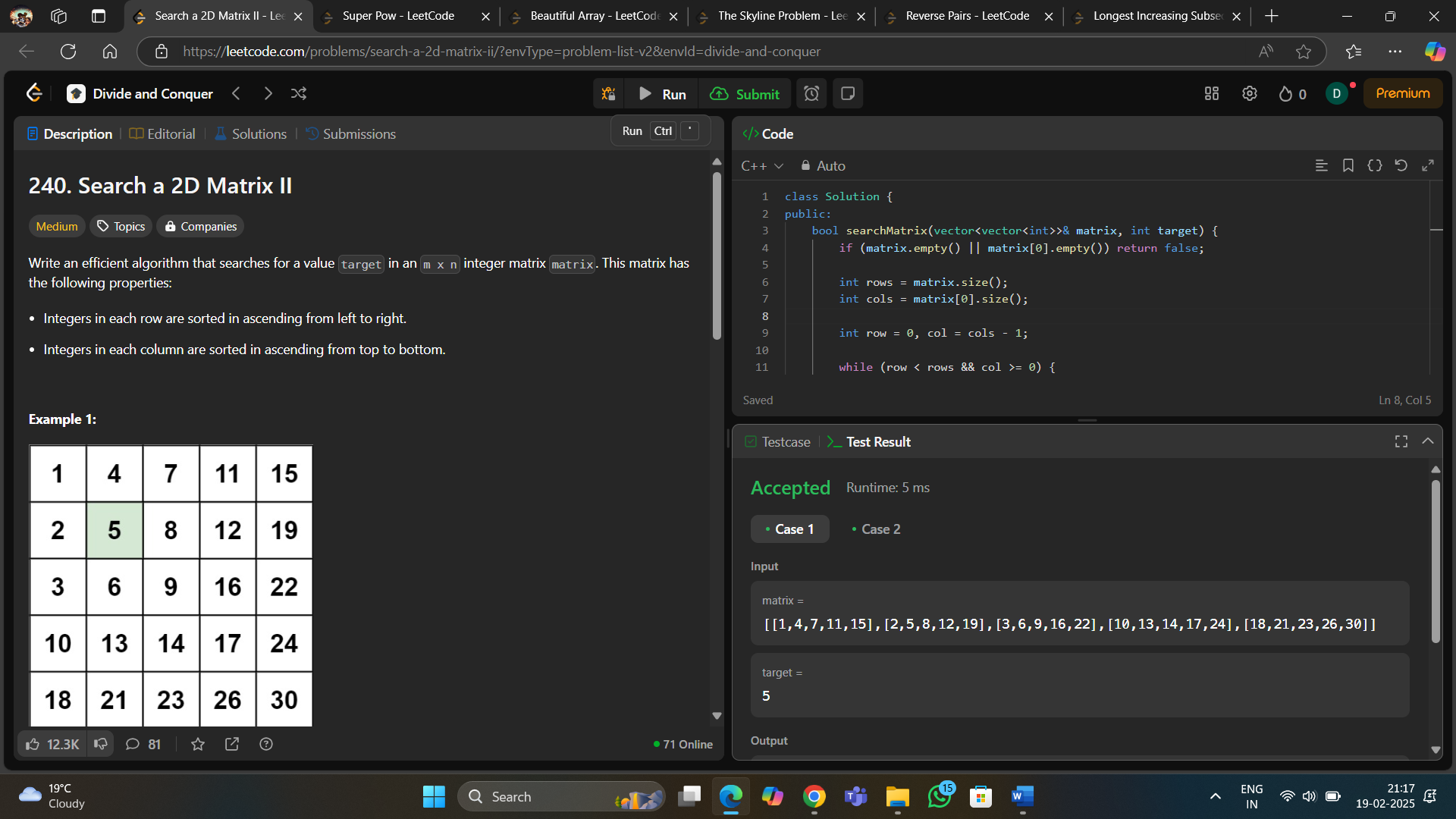
* The subsequence is **strictly increasing** and
* The difference between adjacent elements in the subsequence is **at most** k.

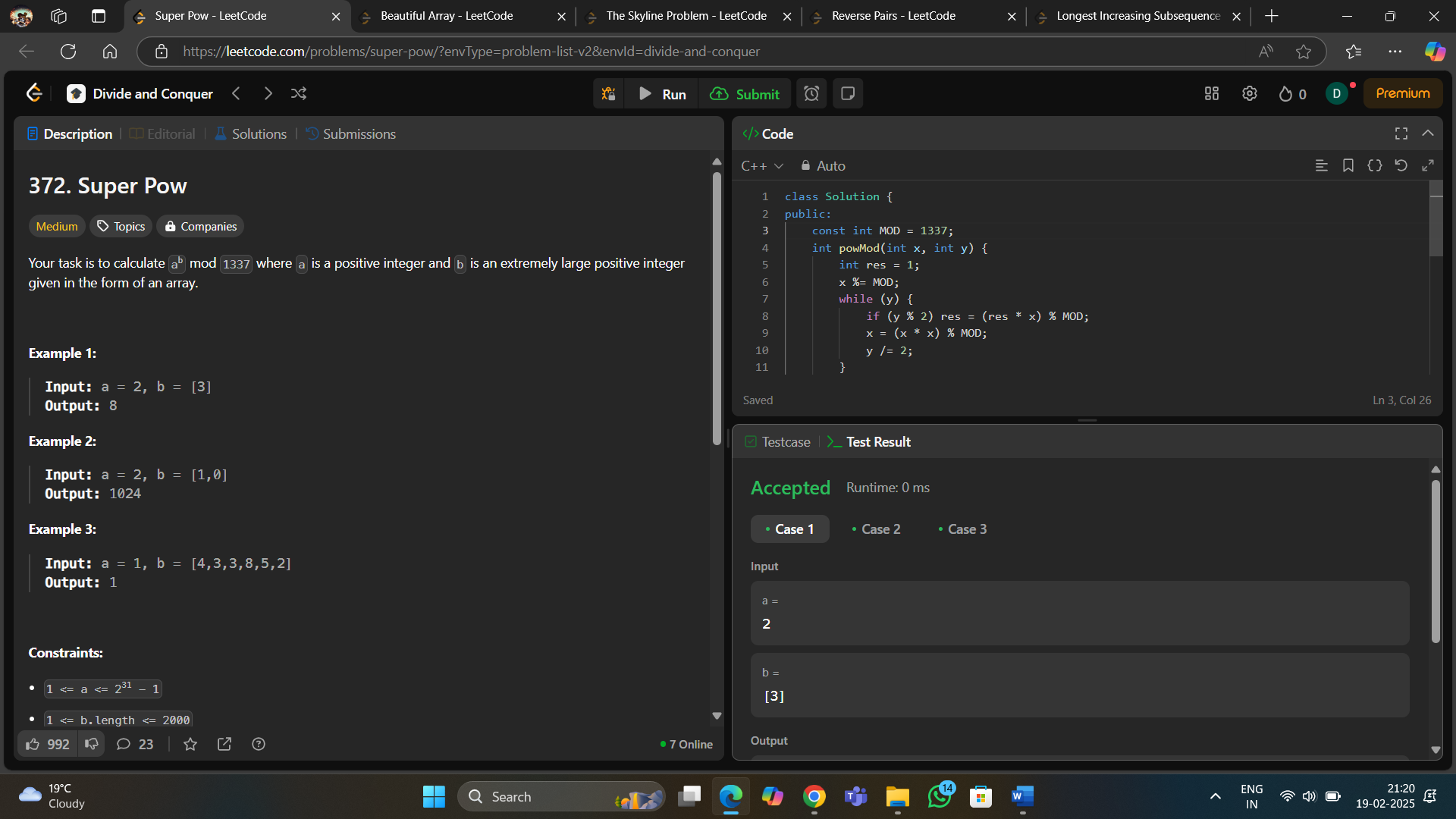
1. **Objective:**

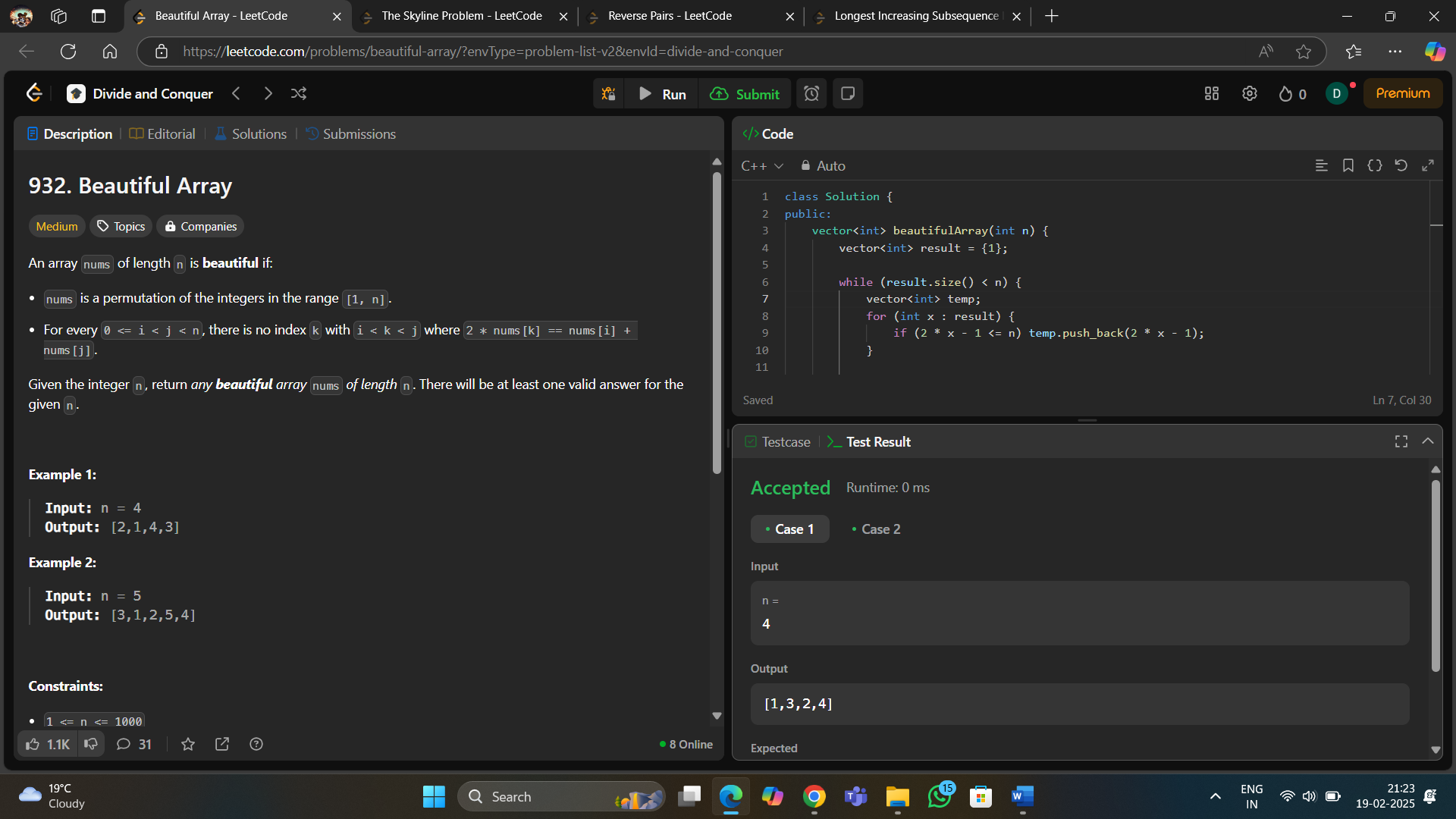
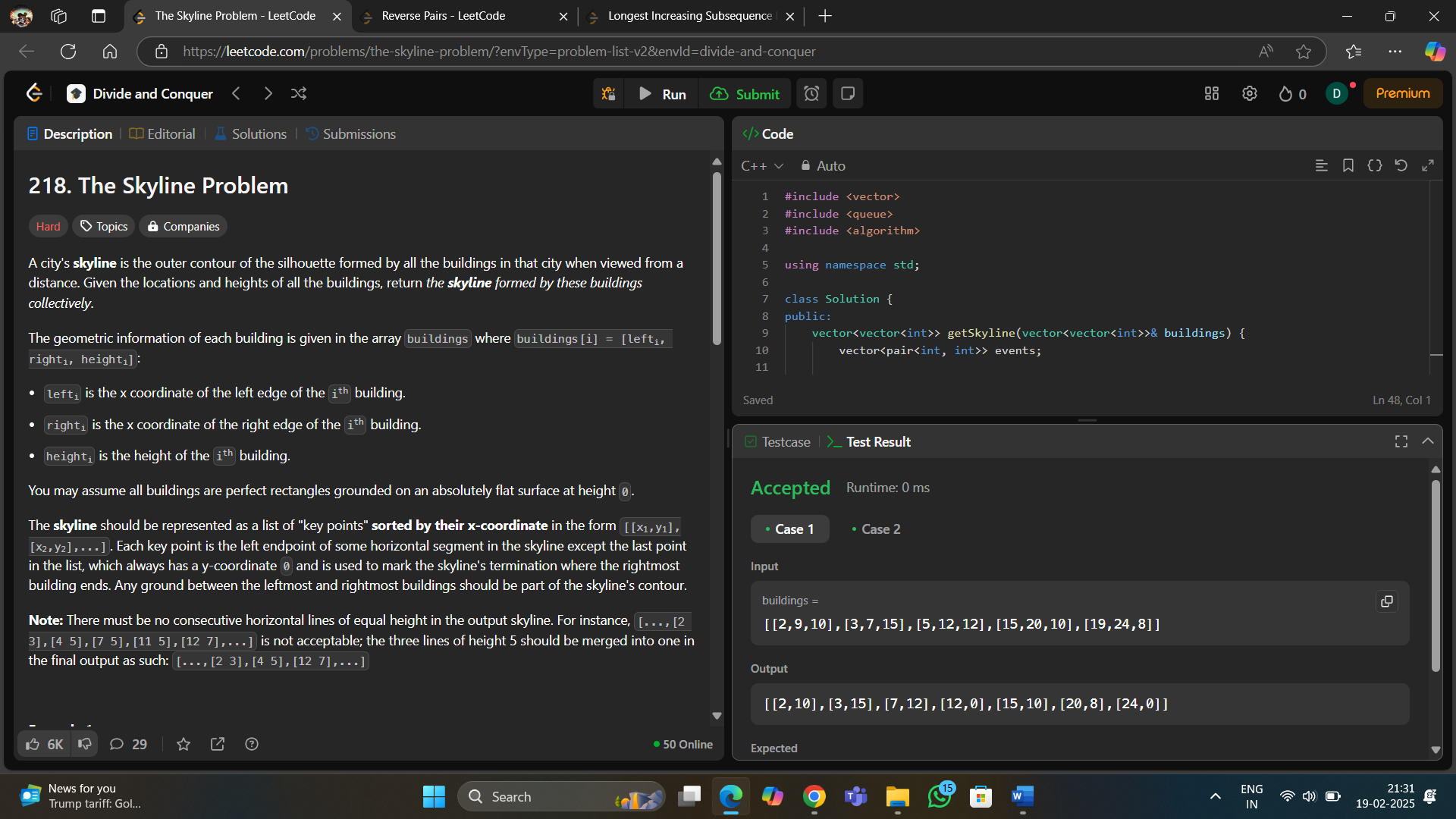
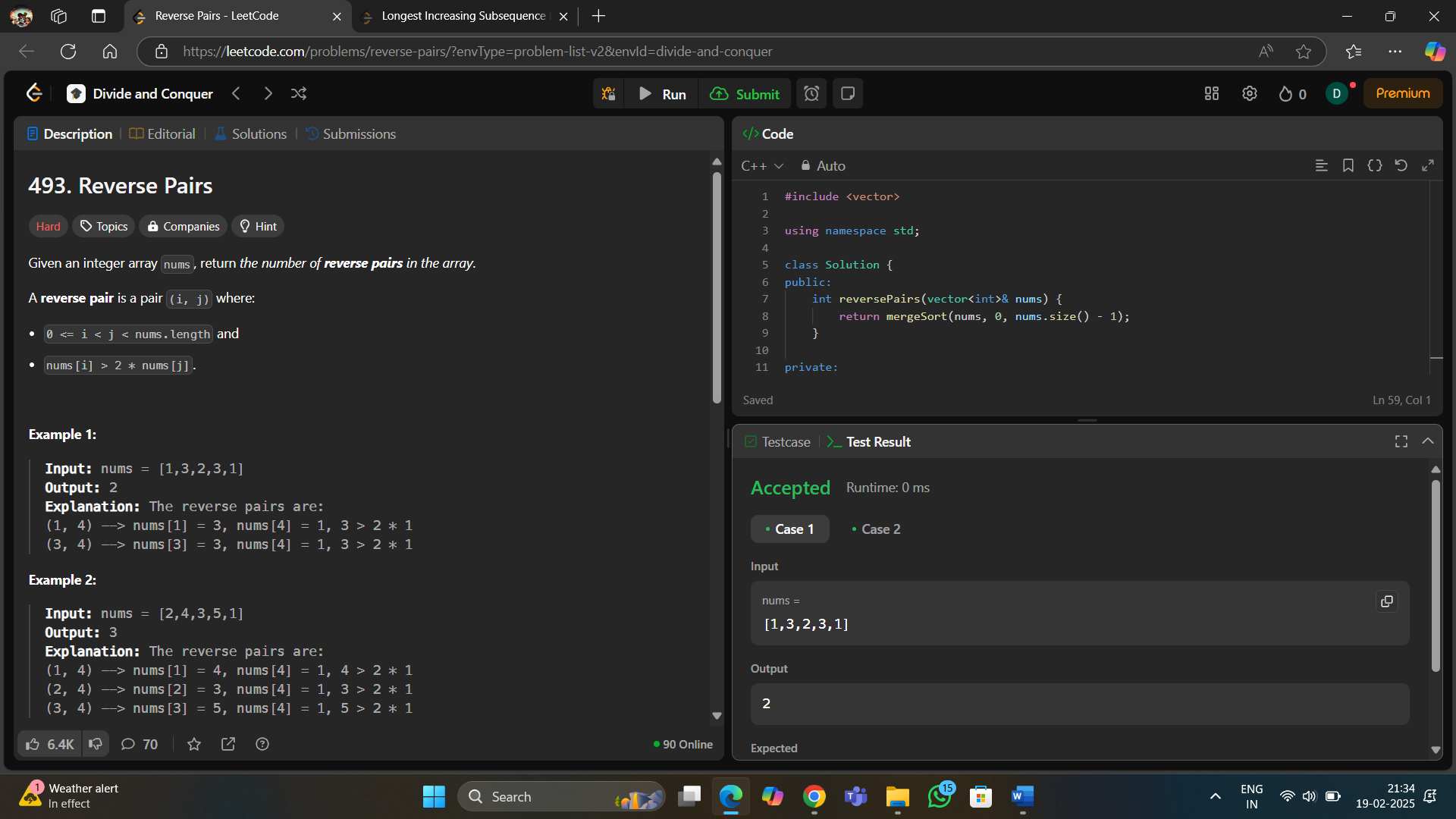
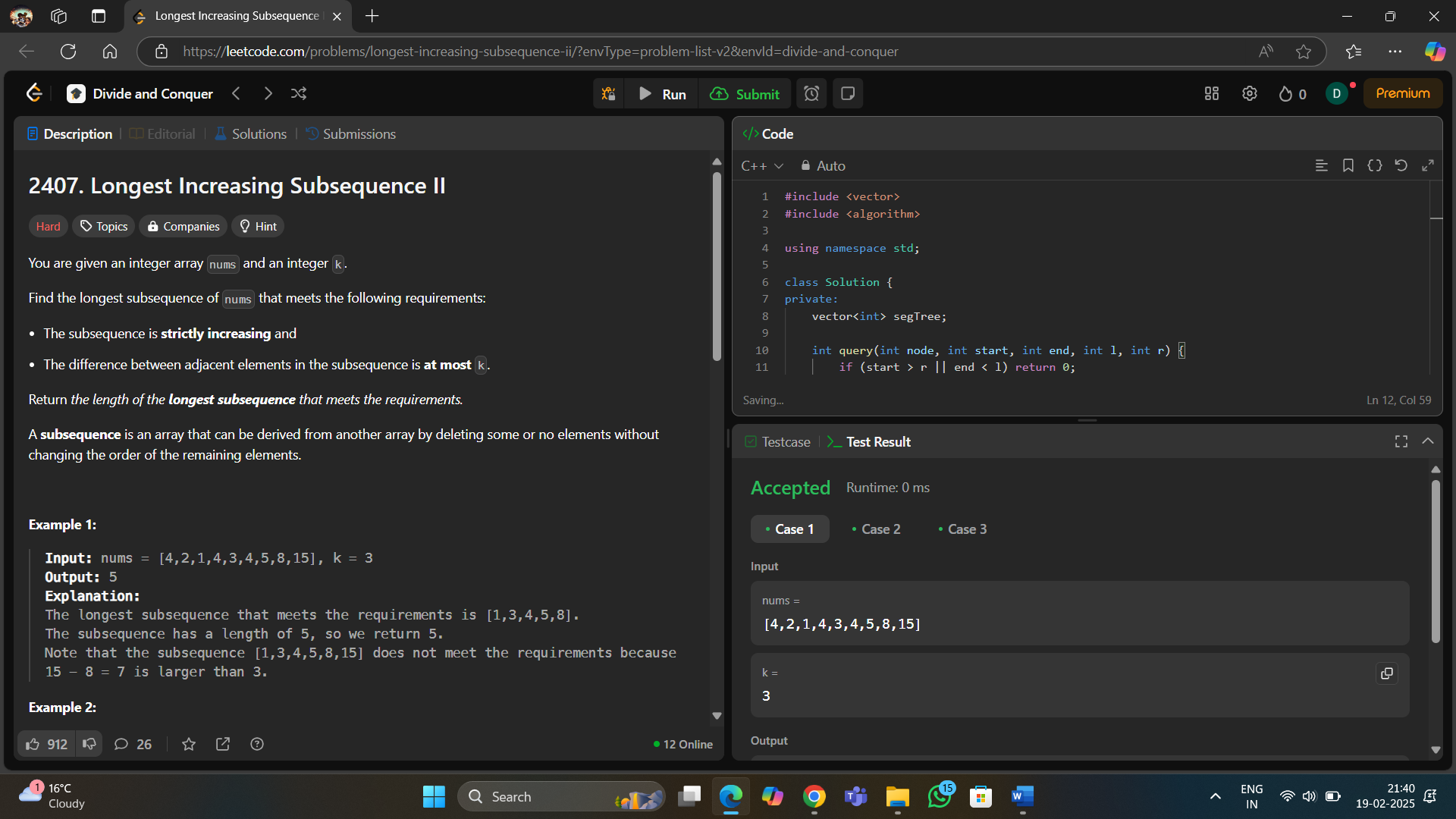
* **Longest Nice Substring** – String manipulation, character set operations.
* **Reverse Bits** – Bitwise operations and binary representation.
* **Number of 1 Bits** – Counting bits, bitwise operations.
* **Maximum Subarray** – Dynamic programming (Kadane’s algorithm).
* **Search a 2D Matrix II** – Binary search and matrix traversal.
* **Super Pow** – Modular exponentiation, recursion, and number theory.
* **Beautiful Array** – Combinatorics, divide and conquer.
* **The Skyline Problem** – Sweep line algorithm, priority queues, computational geometry.
* **Reverse Pairs** – Merge sort, binary indexed trees, segment trees.
* **Longest Increasing Subsequence II** – Dynamic programming, segment trees.

1. **Implementation of Code/Output 1 :**
2. **Code 2 :**

1. **Code 3 :**

1. **Code 4 :**
2. **Code 5 :**
3. **Code 6:**



1. **Code 7 :**
2. **Code 8 :**
3. **Code 9 :**
4. **Code 10 :**
5. **Learning Outcome:**

* Concepts Covered: String Manipulation, Recursion, Divide and Conquer
* Concepts Covered: Bit Manipulation
* Concepts Covered: Bitwise Operations
* Concepts Covered: Dynamic Programming, Greedy Approach
* Concepts Covered: Binary Search, Matrix Search
* Concepts Covered: Modular Arithmetic, Fast Exponentiation
* Concepts Covered: Divide and Conquer, Permutations
* Concepts Covered: Sweep Line Algorithm, Heap (Priority Queue)