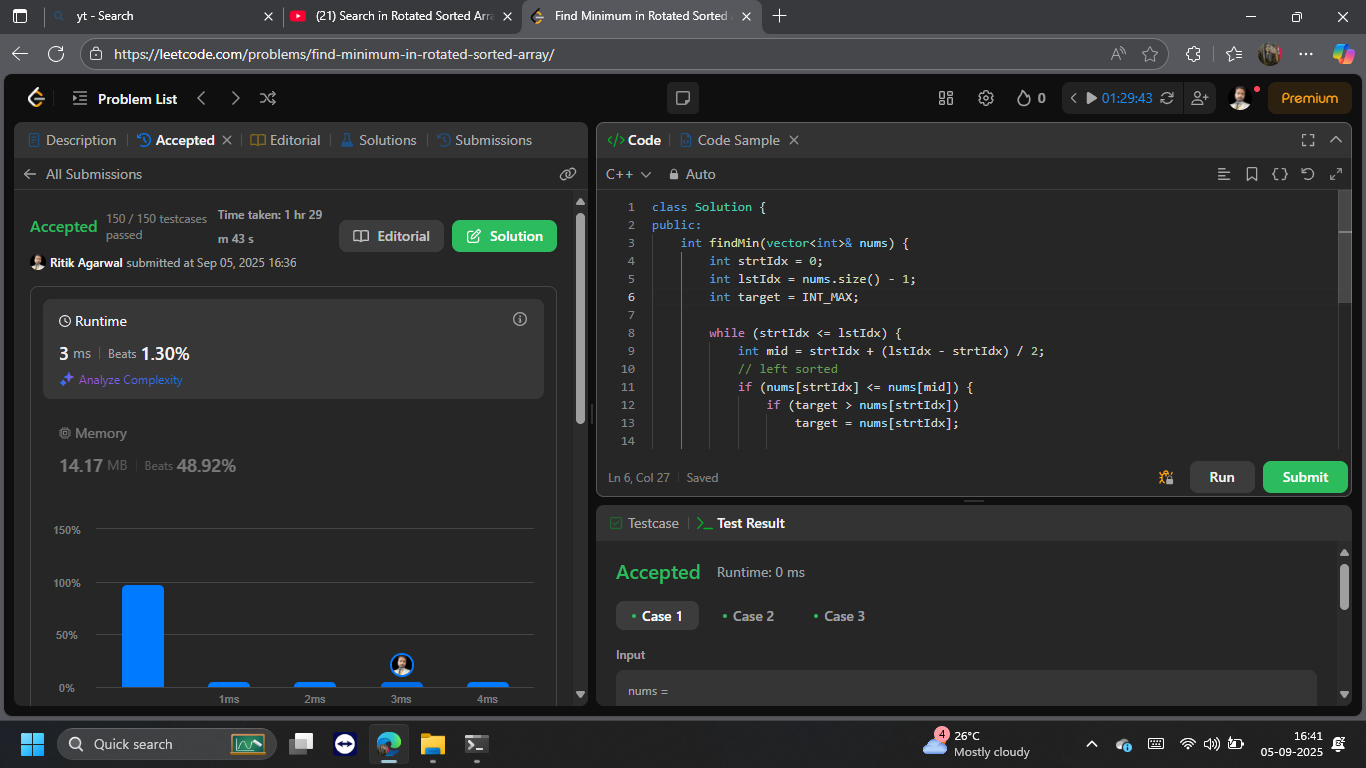
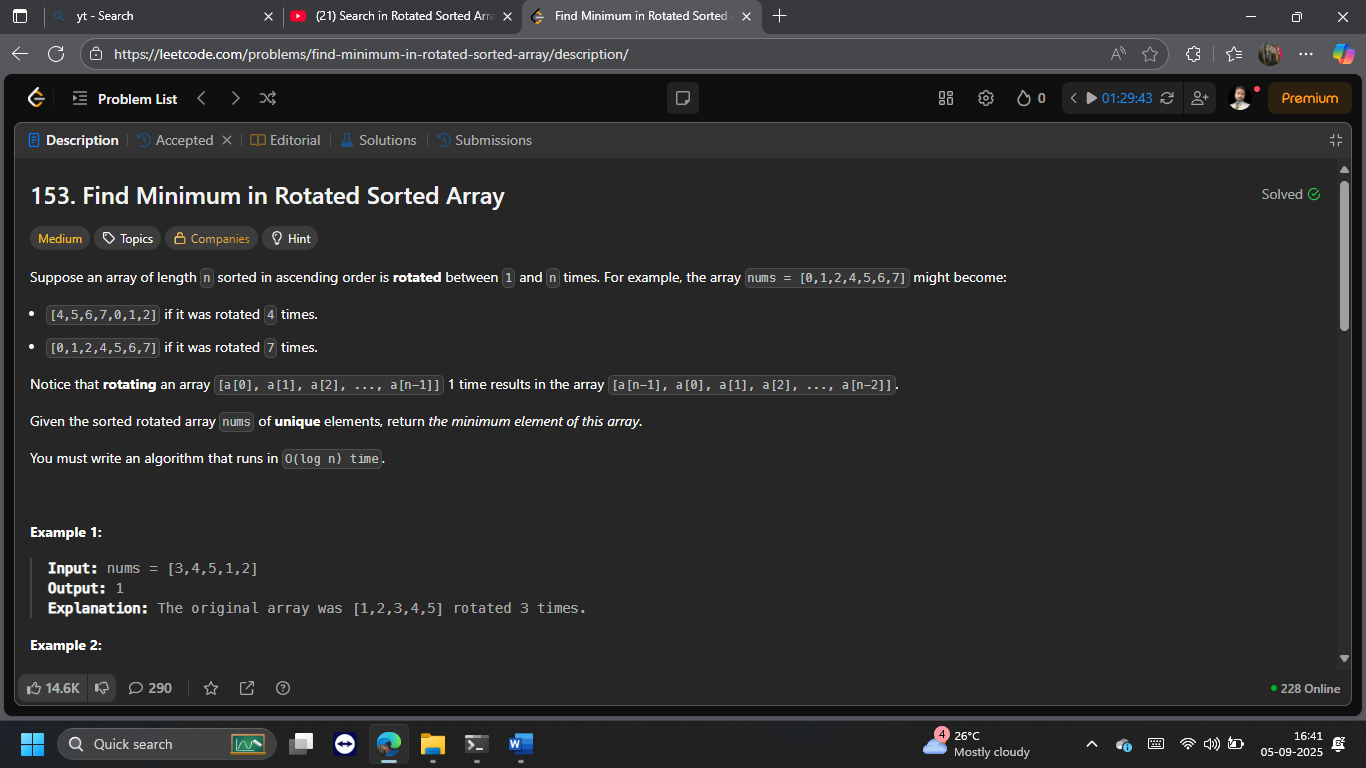
**Leet Code 153 : Find Maximum in a Rotated Sorted Array**

[ Time taken: 1 hr 29 m 43 s ]

the LeetCode problem **“Find Minimum in Rotated Sorted Array”**. This includes the problem statement, your C++ implementation, key learnings, and documentation notes — formatted for clarity and professionalism.

## 🧩 Problem Statement

**Title**: Find Minimum in Rotated Sorted Array  
**Difficulty**: Medium  
**Source**: [LeetCode Problem 153](https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/)

You are given a sorted array of unique integers that has been rotated between 1 and n times. Your task is to find the minimum element in the array.

**Example**:

* Input: nums = [3,4,5,1,2] → Output: 1
* Input: nums = [4,5,6,7,0,1,2] → Output: 0
* Input: nums = [11,13,15,17] → Output: 11

**Constraints**:

* 1 <= nums.length <= 5000
* -5000 <= nums[i] <= 5000
* All elements are unique
* Array is rotated between 1 and n times
* Must run in **O(log n)** time

## 💻 C++ Implementation

class Solution {

public:

int findMin(vector<int>& nums) {

int strtIdx = 0;

int lstIdx = nums.size() - 1;

int target = INT\_MAX;

while (strtIdx <= lstIdx) {

int mid = strtIdx + (lstIdx - strtIdx) / 2;

// Left half is sorted

if (nums[strtIdx] <= nums[mid]) {

if (target > nums[strtIdx])

target = nums[strtIdx];

strtIdx = mid + 1;

}

// Right half is sorted

else {

if (target > nums[mid])

target = nums[mid];

lstIdx = mid - 1;

}

}

return target;

}

};

## 🧠 Key Learnings

### ✅ Binary Search in Rotated Arrays

* A rotated sorted array always has one sorted half.
* By comparing nums[strtIdx] and nums[mid], we can determine which half is sorted.

### ⚠️ Avoiding Pitfalls

* Instead of returning early, this approach tracks the minimum value using a target variable.
* It ensures that both halves are checked for potential minimums.

### 🛠️ Optimization Insight

* This method avoids unnecessary comparisons and branching.
* Maintains **O(log n)** time complexity using binary search principles.

## 📘 Documentation Notes

* This solution is part of my DSA repository for IIIT Bhopal.
* It demonstrates how binary search can be adapted for rotated arrays.
* The use of INT\_MAX as an initial placeholder ensures correctness across all input ranges.
* The logic is clean, readable, and avoids edge-case traps.