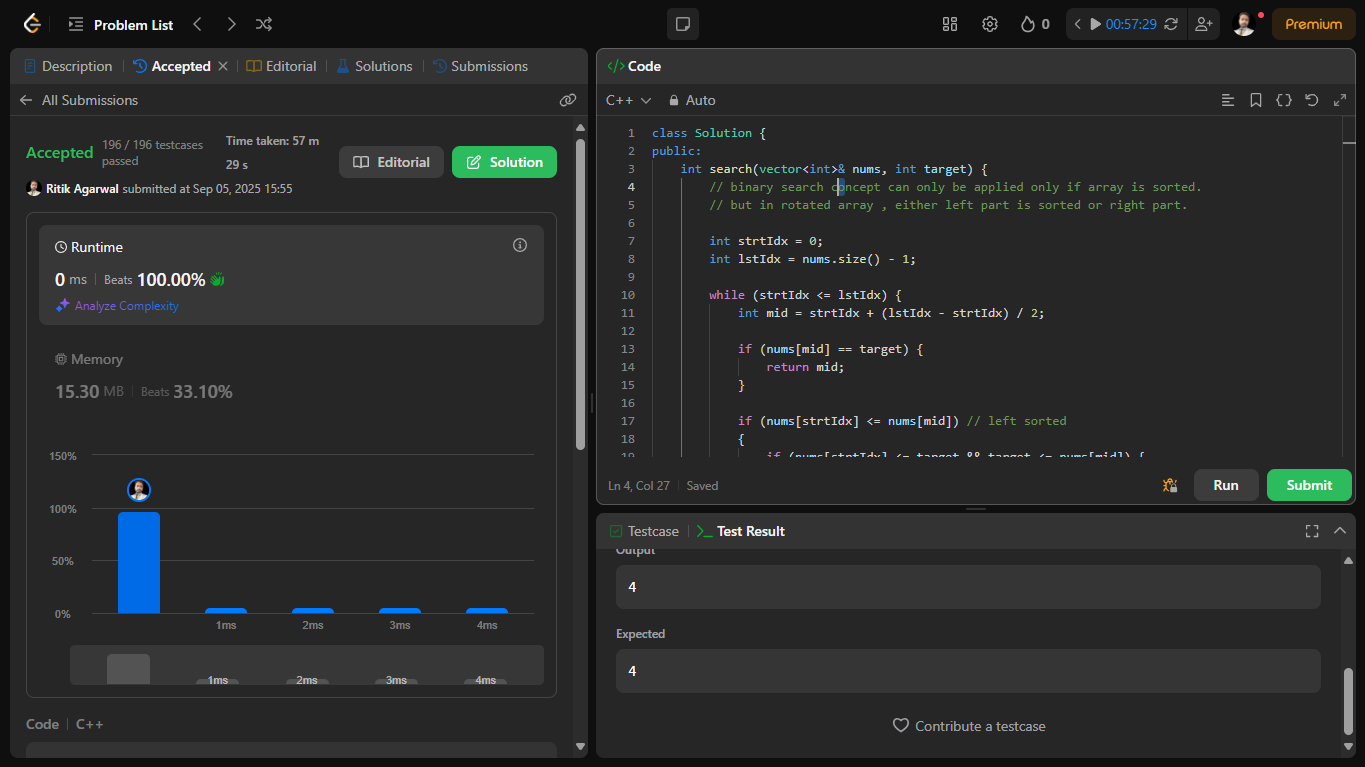
33. Leet Code || Search in a rotated Array



Here’s a structured write-up you can use in your MS Word document to showcase your work on the **“Search in Rotated Sorted Array”** problem from LeetCode:

**🧩 Problem Statement**

**Title**: Search in Rotated Sorted Array  
**Difficulty**: Medium  
**Source**: [LeetCode Problem 33](https://leetcode.com/problems/search-in-rotated-sorted-array/description/)

You are given an integer array nums sorted in ascending order (with distinct values), which may have been rotated at an unknown pivot index k such that the resulting array is:

[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]

Given this array and an integer target, return the index of target if it is in nums, or -1 if it is not.

**Constraints**:

* 1 <= nums.length <= 5000
* -10⁴ <= nums[i], target <= 10⁴
* All values in nums are unique
* Must achieve **O(log n)** time complexity

**🧠 Key Learnings**

**✅ Binary Search Adaptation**

* In a rotated sorted array, **at least one half is always sorted**.
* Use binary search to identify the sorted half and decide which side to search next.

**⚠️ Edge Case Handling**

* The condition nums[start] <= nums[mid] is crucial to correctly identify the sorted half.
* Using < instead of <= can cause incorrect behavior in small arrays like [3, 1].

**🛠️ Debugging Insight**

* A failed test case (nums = [3, 1], target = 1) revealed the importance of inclusive boundary checks.
* Fixing the condition to <= resolved the issue and passed all 196 test cases.

**🧪 Final Implementation (C++)**

class Solution {

public:

int search(vector<int>& nums, int target) {

int start = 0, end = nums.size() - 1;

while (start <= end) {

int mid = start + (end - start) / 2;

if (nums[mid] == target) return mid;

if (nums[start] <= nums[mid]) { // Left half is sorted

if (nums[start] <= target && target < nums[mid])

end = mid - 1;

else

start = mid + 1;

} else { // Right half is sorted

if (nums[mid] < target && target <= nums[end])

start = mid + 1;

else

end = mid - 1;

}

}

return -1;

}

};