Experiment 5

Student Name: Payal Singh **UID:22BET10347**
Branch: BE-IT **Section/Group: IOT-702(A)**
Semester: 6 **Date of Performance:21/02/25**
Subject Name: AP LAB-II **Subject Code: 22ITP-351**

PROBLEM 1:

Aim:

you are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively.

Merge nums1 and nums2 into a single array sorted in non-decreasing order.

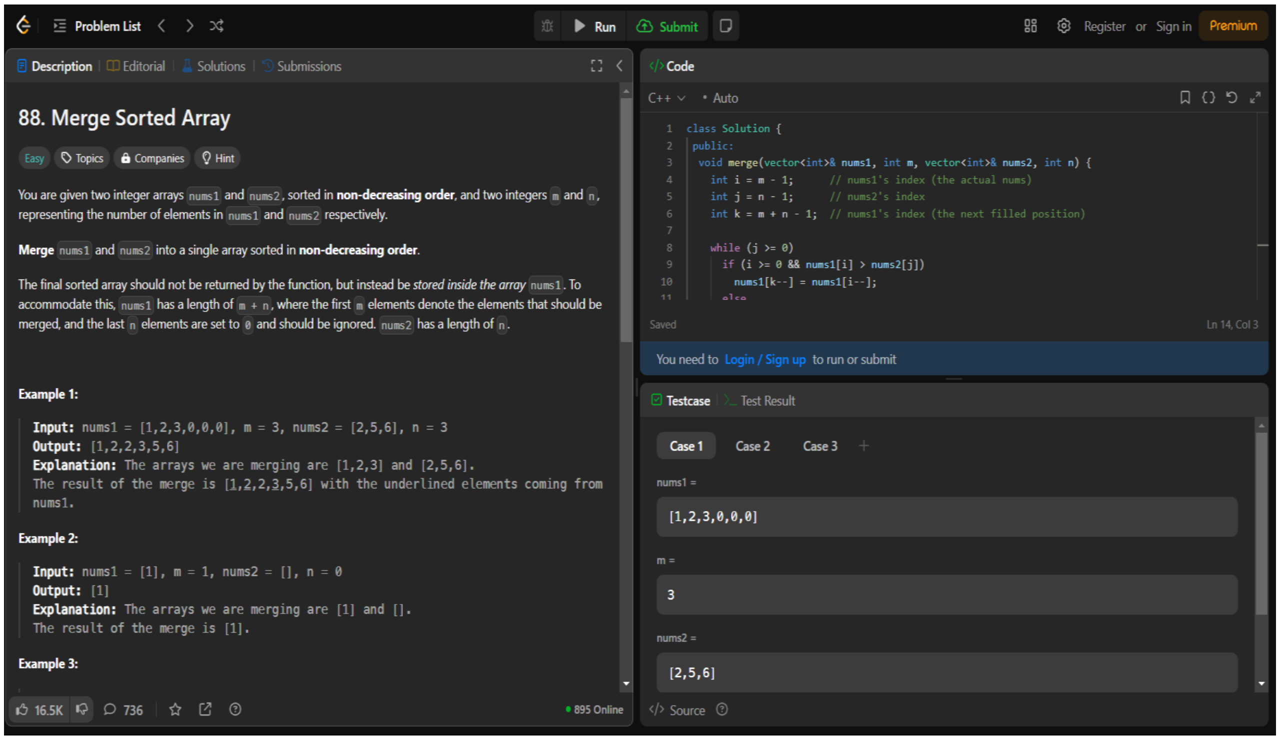
The final sorted array should not be returned by the function, but instead be stored inside the array nums1. To accommodate this, nums1 has a length of m + n, where the first m elements denote the elements that should be merged, and the last n elements are set to 0 and should be ignored. nums2 has a length of n.

Code:

```
class Solution {
 public:
  void
merge(vector<int>&
nums1, int m,
vector<int>&
nums2, int n) {
    int i = m - 1;
    int j = n - 1;
    int k = m + n - 1;
while (j >= 0)
      if (i >= 0 &&
nums1[i] >
nums2[j])
        nums1[k--] =
nums1[i--];
      else
        nums1[k--] =
nums2[j--];
```

```
    }
  };
```

Output:



88. Merge Sorted Array

You are given two integer arrays `nums1` and `nums2`, sorted in **non-decreasing order**, and two integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively.

Merge `nums1` and `nums2` into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function, but instead be *stored inside the array* `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elements denote the elements that should be merged, and the last `n` elements are set to `0` and should be ignored. `nums2` has a length of `n`.

Example 1:

```
Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3
Output: [1,2,2,3,5,6]
Explanation: The arrays we are merging are [1,2,3] and [2,5,6].
The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.
```

Example 2:

```
Input: nums1 = [1], m = 1, nums2 = [], n = 0
Output: [1]
Explanation: The arrays we are merging are [1] and [].
The result of the merge is [1].
```

Example 3:

```cpp
class Solution {
 public:
  void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
    int i = m - 1;      // nums1's index (the actual nums)
    int j = n - 1;      // nums2's index
    int k = m + n - 1;  // nums1's index (the next filled position)

    while (j >= 0)
      if (i >= 0 && nums1[i] > nums2[j])
        nums1[k--] = nums1[i--];
      else
```

Saved   Ln 14, Col 3

You need to Login / Sign up to run or submit

Testcase | Test Result

Case 1 Case 2 Case 3

nums1 =
```
[1,2,3,0,0,0]
```
m =
```
3
```
nums2 =
```
[2,5,6]
```

PROBLEM 2:

Aim: You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.

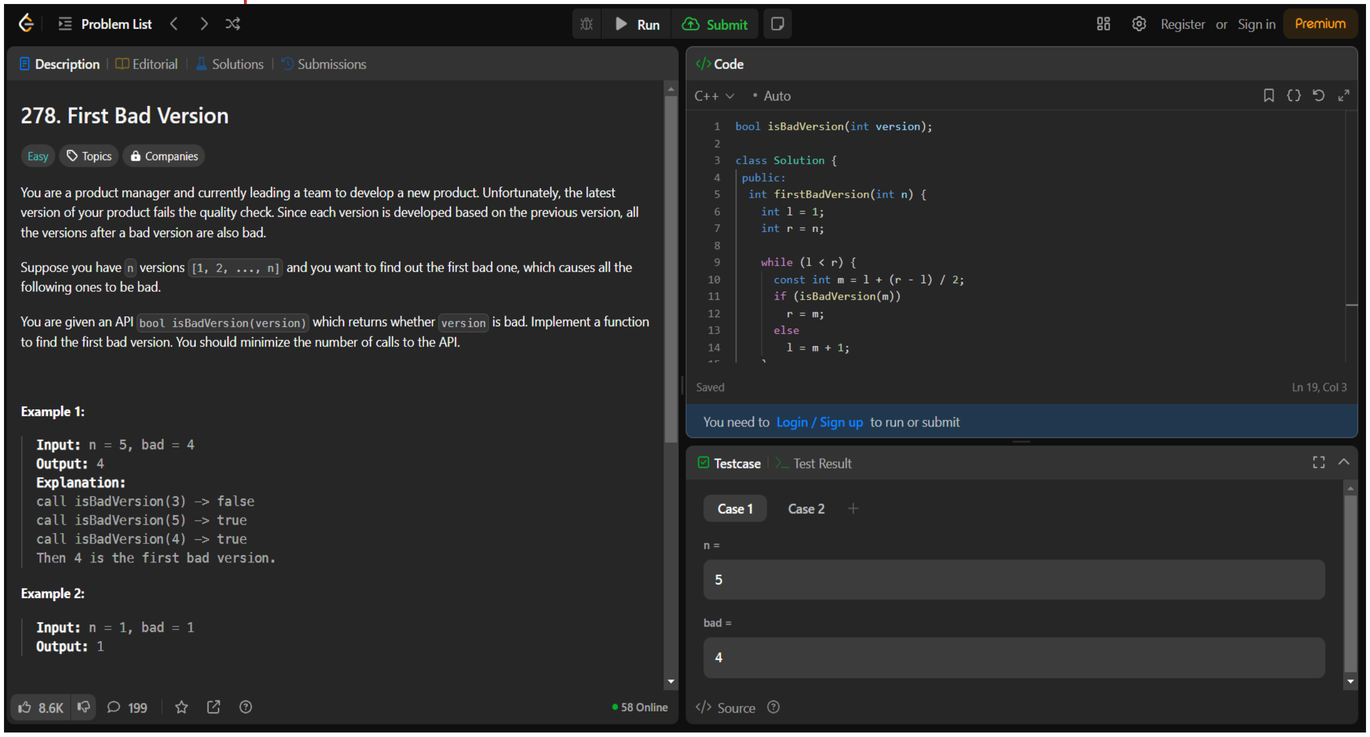
Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes all the following ones to be bad.

You are given an API bool isBadVersion(version) which returns whether version is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.
.

Code:

```
bool isBadVersion(int version);
class Solution {
 public:
  int firstBadVersion(int n) {
    int l = 1;
    int r = n;

    while (l < r) {
      const int m = l + (r - l) / 2;
      if (isBadVersion(m))
        r = m;
      else
        l = m + 1;
    }

    return l;
  }
};
```

Output:



278. First Bad Version

Easy · Topics · Companies

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.

Suppose you have `n` versions `[1, 2, ..., n]` and you want to find out the first bad one, which causes all the following ones to be bad.

You are given an API `bool isBadVersion(version)` which returns whether `version` is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

Example 1:

```
Input: n = 5, bad = 4
Output: 4
Explanation:
call isBadVersion(3) -> false
call isBadVersion(5) -> true
call isBadVersion(4) -> true
Then 4 is the first bad version.
```

Example 2:

```
Input: n = 1, bad = 1
Output: 1
```

8.6K · 199 · 58 Online

```
bool isBadVersion(int version);

class Solution {
 public:
  int firstBadVersion(int n) {
    int l = 1;
    int r = n;

    while (l < r) {
      const int m = l + (r - l) / 2;
      if (isBadVersion(m))
        r = m;
      else
        l = m + 1;
```

Saved   Ln 19, Col 3

You need to Login / Sign up to run or submit

Testcase | Test Result

Case 1   Case 2

n =
5

bad =
4

Source

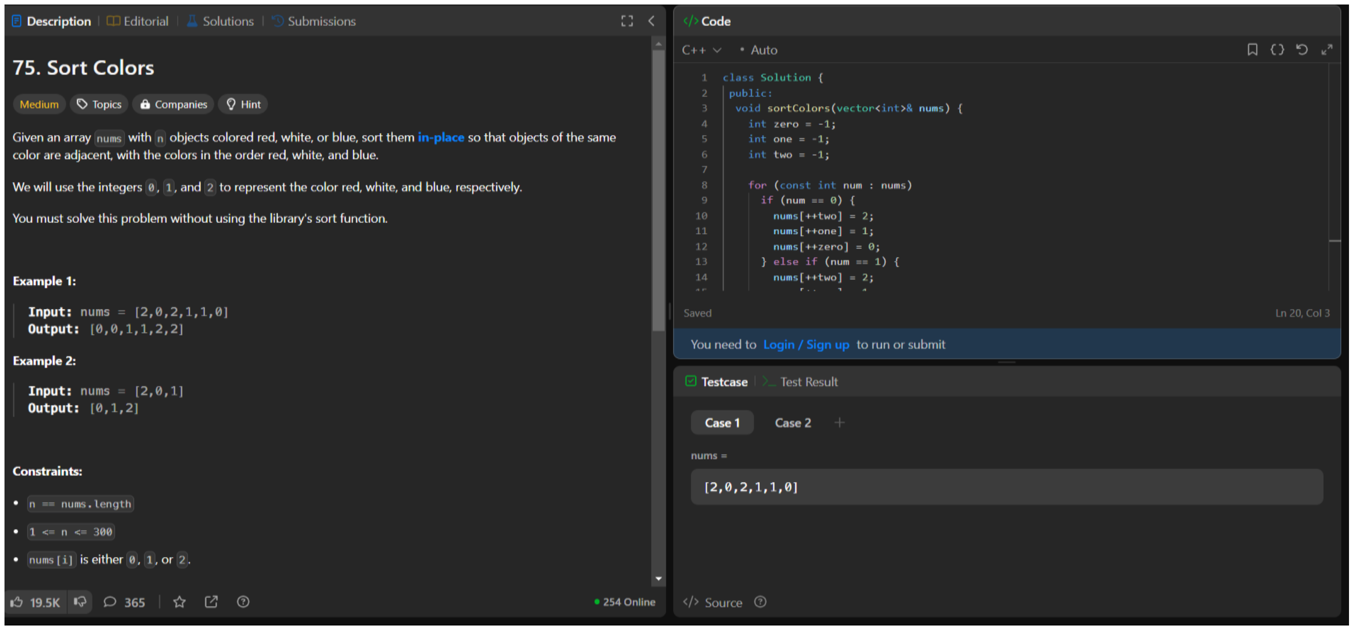
PROBLEM 3:

Aim: **Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue.**
We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.
You must solve this problem without using the library's sort function.

Code:

```
class Solution {
 public:
  void sortColors(vector<int>& nums) {
    int zero = -1;
    int one = -1;
    int two = -1;

    for (const int num : nums)
      if (num == 0) {
        nums[++two] = 2;
        nums[++one] = 1;
        nums[++zero] = 0;
      } else if (num == 1) {
        nums[++two] = 2;
        nums[++one] = 1;
      } else {
        nums[++two] = 2;
      }
    }
  }
};
```

Output:



Description | Editorial | Solutions | Submissions

75. Sort Colors

Medium | Topics | Companies | Hint

Given an array `nums` with `n` objects colored red, white, or blue, sort them **in-place** so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers `0`, `1`, and `2` to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

```
Input: nums = [2,0,2,1,1,0]
Output: [0,0,1,1,2,2]
```

Example 2:

```
Input: nums = [2,0,1]
Output: [0,1,2]
```

Constraints:

- `n == nums.length`
- `1 <= n <= 300`
- `nums[i]` is either `0`, `1`, or `2`.

19.5K | 365 | 254 Online

Code

C++ · Auto

```
class Solution {
 public:
  void sortColors(vector<int>& nums) {
    int zero = -1;
    int one = -1;
    int two = -1;

    for (const int num : nums)
      if (num == 0) {
        nums[++two] = 2;
        nums[++one] = 1;
        nums[++zero] = 0;
      } else if (num == 1) {
        nums[++two] = 2;
```

Saved | Ln 20, Col 3

You need to Login / Sign up to run or submit

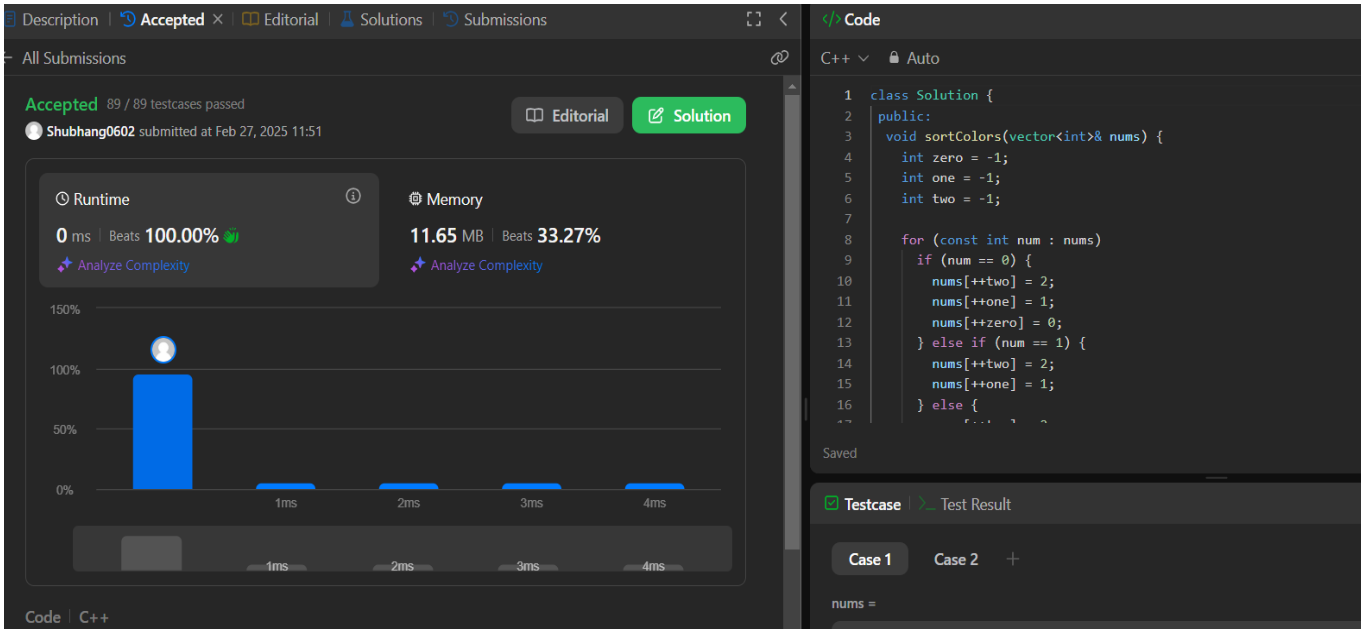
Testcase | Test Result

Case 1 | Case 2

nums =

```
[2,0,2,1,1,0]
```

Source



Accepted 89 / 89 testcases passed
Shubhang0602 submitted at Feb 27, 2025 11:51

Runtime
0 ms | Beats 100.00%

Memory
11.65 MB | Beats 33.27%

```
class Solution {
public:
  void sortColors(vector<int>& nums) {
    int zero = -1;
    int one = -1;
    int two = -1;

    for (const int num : nums)
      if (num == 0) {
        nums[++two] = 2;
        nums[++one] = 1;
        nums[++zero] = 0;
      } else if (num == 1) {
        nums[++two] = 2;
        nums[++one] = 1;
      } else {
```

Saved

nums =

PROBLEM 4:

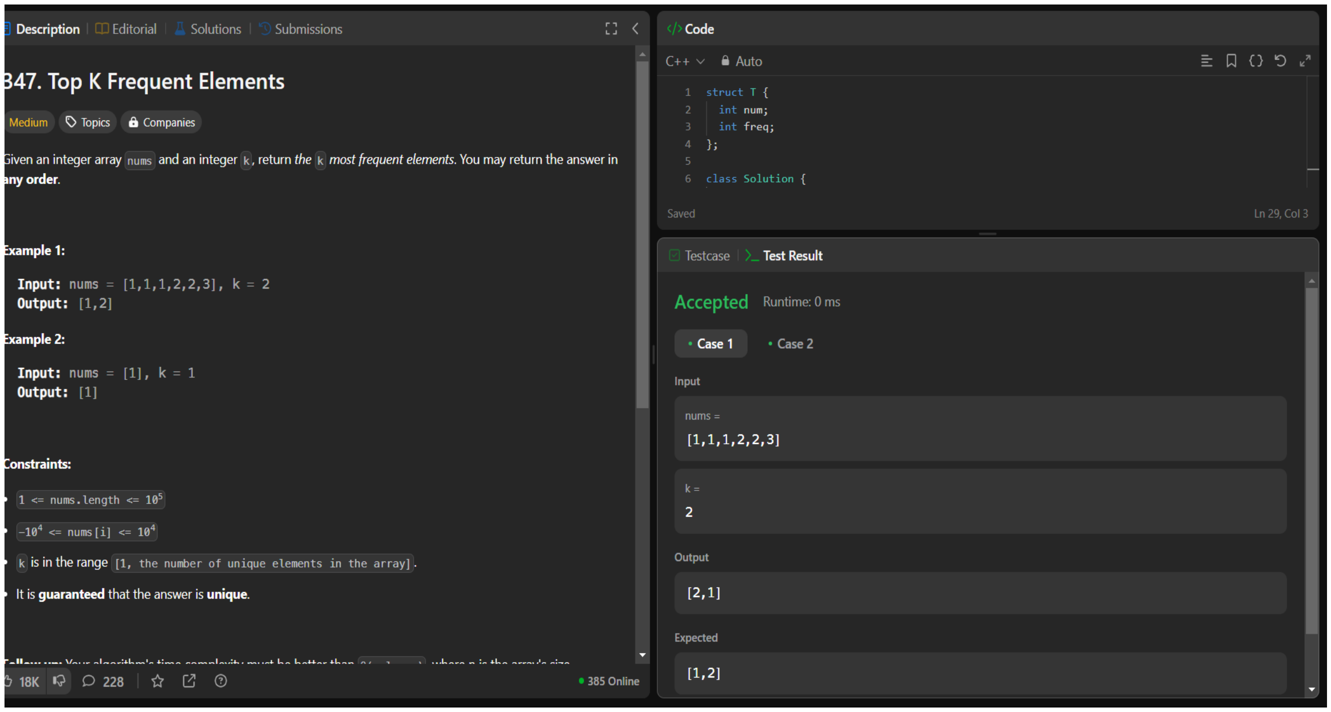
Aim: Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order.

Code:

```
struct T {
  int num;
  int freq;
};

class Solution {
 public:
  vector<int> topKFrequent(vector<int>& nums, int k) {
    const int n = nums.size();
    vector<int> ans;
    unordered_map<int, int> count;
    auto compare = [](const T& a, const T& b) { return a.freq > b.freq; };
    priority_queue<T, vector<T>, decltype(compare)> minHeap(compare);

    for (const int num : nums)
      ++count[num];

    for (const auto& [num, freq] : count) {
      minHeap.emplace(num, freq);
      if (minHeap.size() > k)
        minHeap.pop();
    }

    while (!minHeap.empty())
```

```
        ans.push_back(minHeap.top().num), minHeap.pop();

      return ans;
    }
};
```

OUTPUT:



Description | Editorial | Solutions | Submissions

347. Top K Frequent Elements

Medium · Topics · Companies

Given an integer array `nums` and an integer `k`, return the `k` *most frequent elements*. You may return the answer in **any order**.

Example 1:

```
Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]
```

Example 2:

```
Input: nums = [1], k = 1
Output: [1]
```

Constraints:

- `1 <= nums.length <= 10^5`
- `-10^4 <= nums[i] <= 10^4`
- `k` is in the range `[1, the number of unique elements in the array]`.
- It is **guaranteed** that the answer is **unique**.

18K · 228 · 385 Online

Code — C++ · Auto

```
struct T {
  int num;
  int freq;
};

class Solution {
```

Saved · Ln 29, Col 3

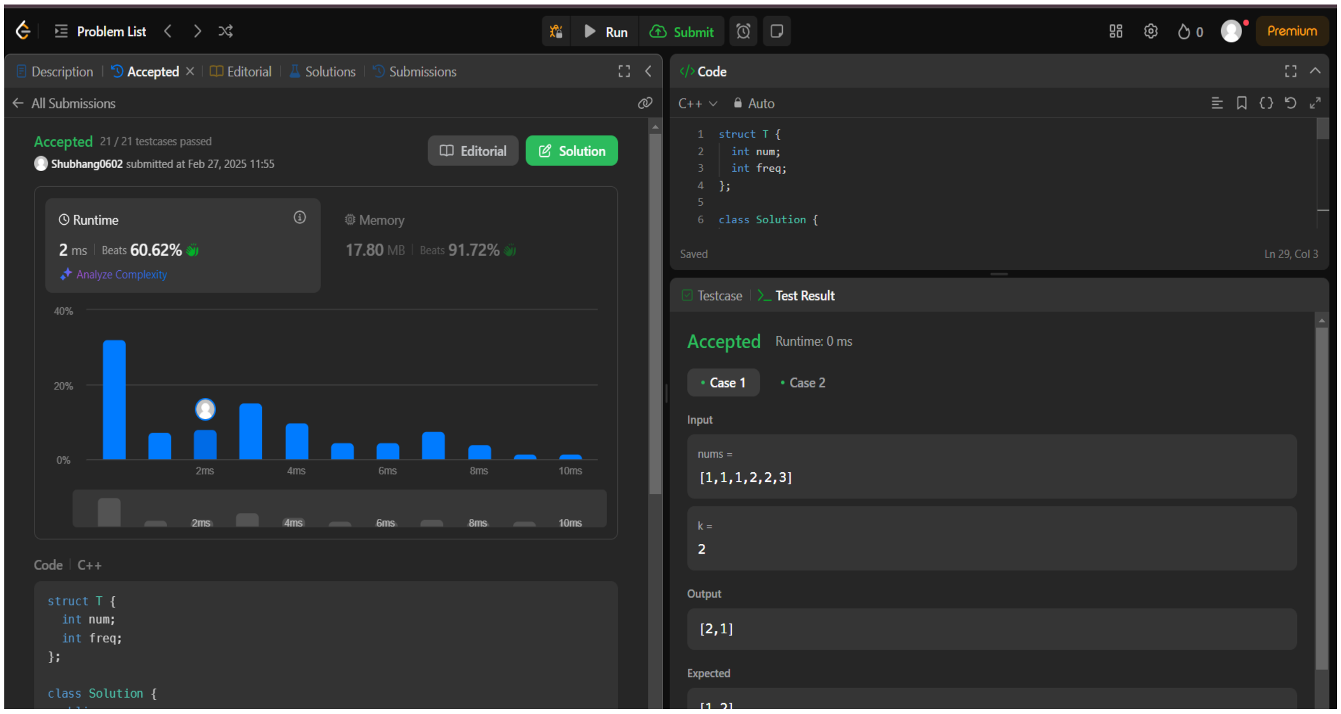
Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

nums =
```
[1,1,1,2,2,3]
```
k =
```
2
```
Output
```
[2,1]
```
Expected
```
[1,2]
```



Problem List · Run · Submit · Premium

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 21 / 21 testcases passed

Shubhang0602 submitted at Feb 27, 2025 11:55

Editorial · Solution

Runtime: 2 ms Beats 60.62%

Analyze Complexity

Memory: 17.80 MB Beats 91.72%

Code | C++

```
struct T {
  int num;
  int freq;
};

class Solution {
```

Code — C++ · Auto

```
struct T {
  int num;
  int freq;
};

class Solution {
```

Saved · Ln 29, Col 3

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

nums =
```
[1,1,1,2,2,3]
```
k =
```
2
```
Output
```
[2,1]
```
Expected

PROBLEM 5:

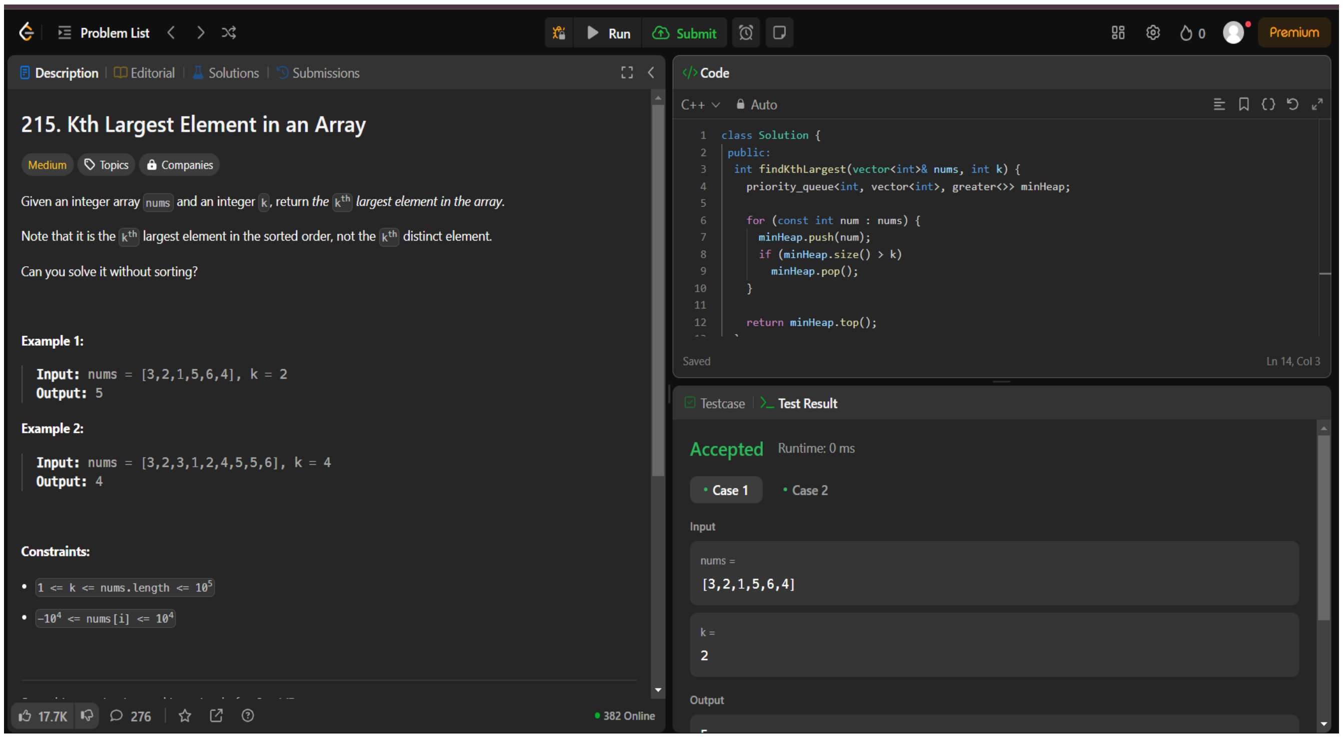
Aim: Given an integer array nums and an integer k, return the kth largest element in the array.

**Note that it is the kth largest element in the sorted order, not the kth distinct element.
Can you solve it without sorting?**

Code:

```
class Solution {
 public:
  int findKthLargest(vector<int>& nums, int k) {
    priority_queue<int, vector<int>, greater<>> minHeap;

    for (const int num : nums) {
      minHeap.push(num);
      if (minHeap.size() > k)
        minHeap.pop();
    }

    return minHeap.top();
  }
};
```

Output:



Problem List · Run · Submit · Premium

Description | Editorial | Solutions | Submissions

215. Kth Largest Element in an Array

Medium · Topics · Companies

Given an integer array `nums` and an integer `k`, return *the* `k`th *largest element in the array*.

Note that it is the `k`th largest element in the sorted order, not the `k`th distinct element.

Can you solve it without sorting?

Example 1:

```
Input: nums = [3,2,1,5,6,4], k = 2
Output: 5
```

Example 2:

```
Input: nums = [3,2,3,1,2,4,5,5,6], k = 4
Output: 4
```

Constraints:

- `1 <= k <= nums.length <= 10`^{5}
- `-10`^{4} `<= nums[i] <= 10`^{4}

17.7K · 276 · 382 Online

Code · C++ · Auto

```
class Solution {
 public:
  int findKthLargest(vector<int>& nums, int k) {
    priority_queue<int, vector<int>, greater<>> minHeap;

    for (const int num : nums) {
      minHeap.push(num);
      if (minHeap.size() > k)
        minHeap.pop();
    }

    return minHeap.top();
```

Saved · Ln 14, Col 3

Testcase | Test Result

Accepted   Runtime: 0 ms

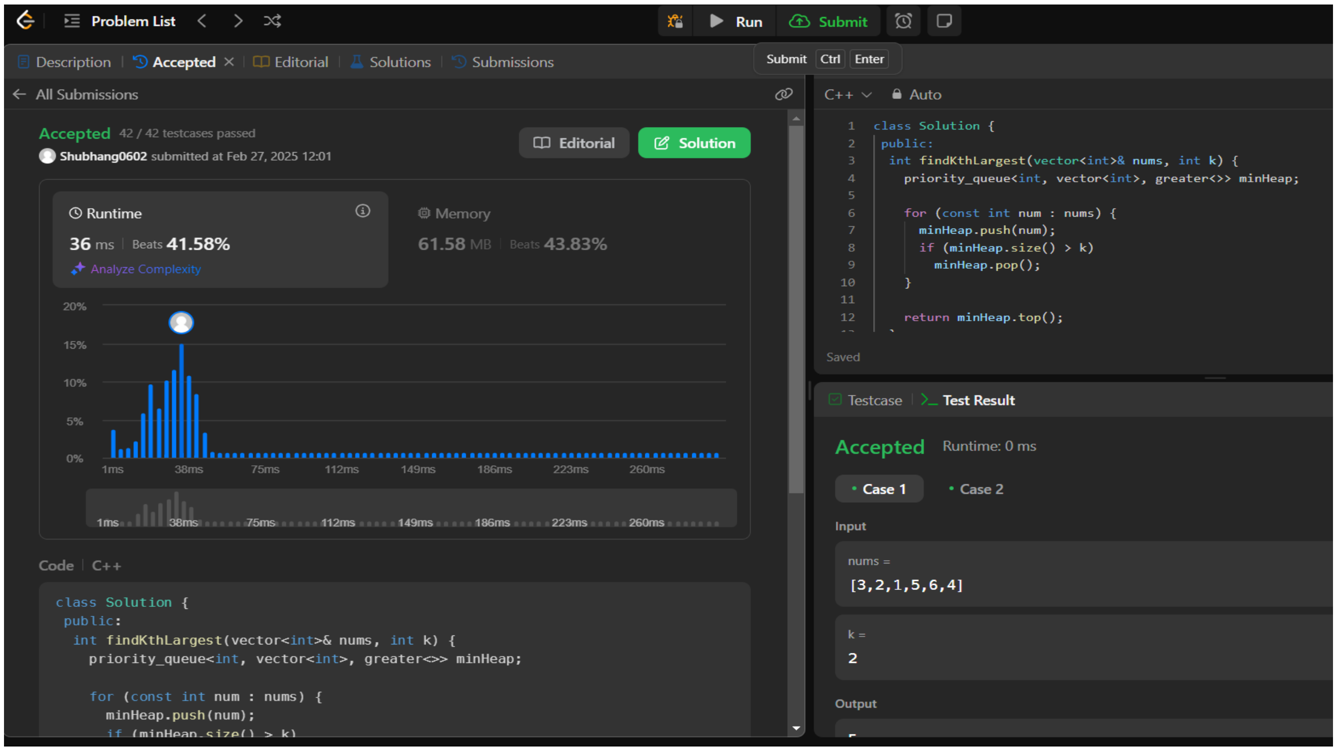
Case 1 · Case 2

Input

nums =
[3,2,1,5,6,4]

k =
2

Output



Accepted 42 / 42 testcases passed
Shubhang0602 submitted at Feb 27, 2025 12:01

Runtime
36 ms | Beats 41.58%

Memory
61.58 MB | Beats 43.83%

```
class Solution {
 public:
  int findKthLargest(vector<int>& nums, int k) {
    priority_queue<int, vector<int>, greater<>> minHeap;

    for (const int num : nums) {
      minHeap.push(num);
      if (minHeap.size() > k)
        minHeap.pop();
    }

    return minHeap.top();
```

Accepted Runtime: 0 ms

Input

nums =
[3,2,1,5,6,4]

k =
2

Output

PROBLEM 6:

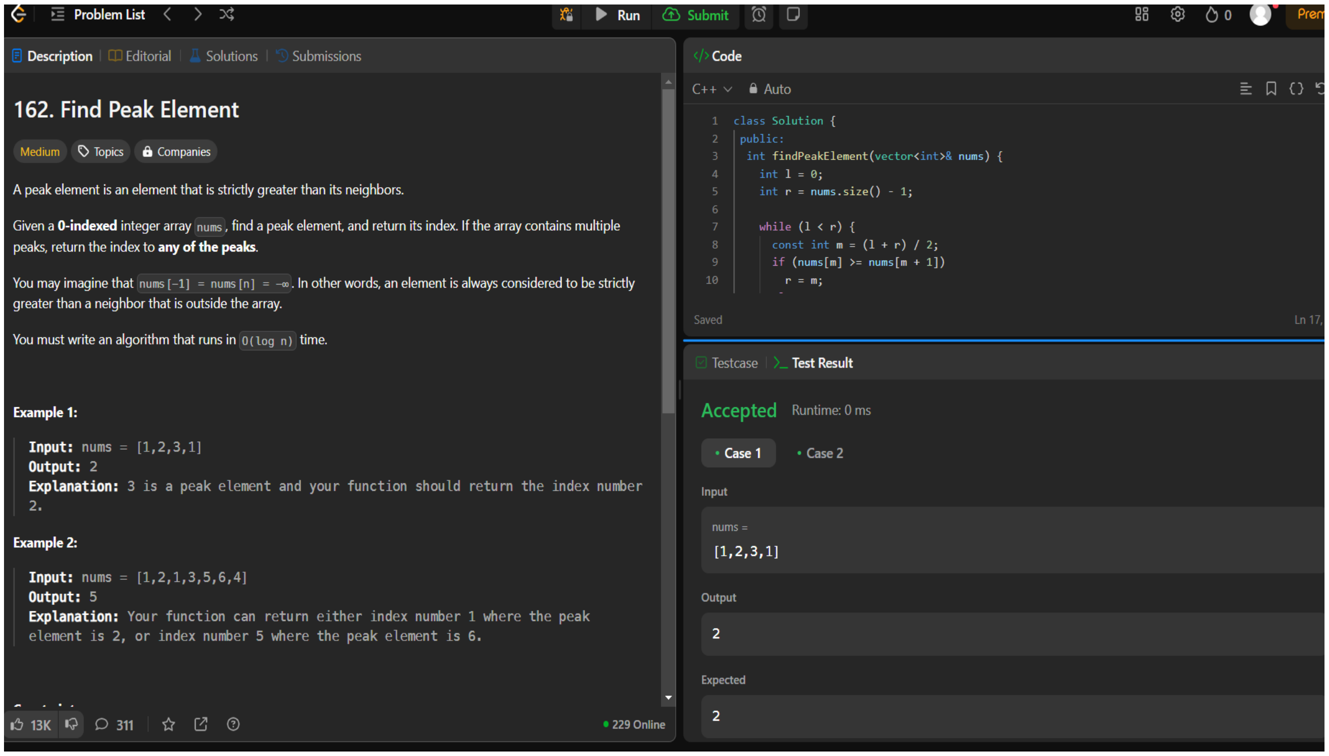
Aim: A peak element is an element that is strictly greater than its neighbors.

- **Given a 0-indexed integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to any of the peaks.**
- **You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.**
- **You must write an algorithm that runs in O(log n) time.**

Code:

```
class Solution {
 public:
  int findPeakElement(vector<int>& nums) {
    int l = 0;
    int r = nums.size() - 1;

    while (l < r) {
      const int m = (l + r) / 2;
      if (nums[m] >= nums[m + 1])
        r = m;
      else
        l = m + 1;
    }

    return l;
  }
};
```

Output:



162. Find Peak Element

Medium · Topics · Companies

A peak element is an element that is strictly greater than its neighbors.

Given a **0-indexed** integer array `nums`, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that `nums[-1] = nums[n] = -∞`. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in `O(log n)` time.

Example 1:

```
Input: nums = [1,2,3,1]
Output: 2
Explanation: 3 is a peak element and your function should return the index number 2.
```

Example 2:

```
Input: nums = [1,2,1,3,5,6,4]
Output: 5
Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.
```

13K · 311 · 229 Online

```cpp
class Solution {
public:
  int findPeakElement(vector<int>& nums) {
    int l = 0;
    int r = nums.size() - 1;

    while (l < r) {
      const int m = (l + r) / 2;
      if (nums[m] >= nums[m + 1])
        r = m;
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

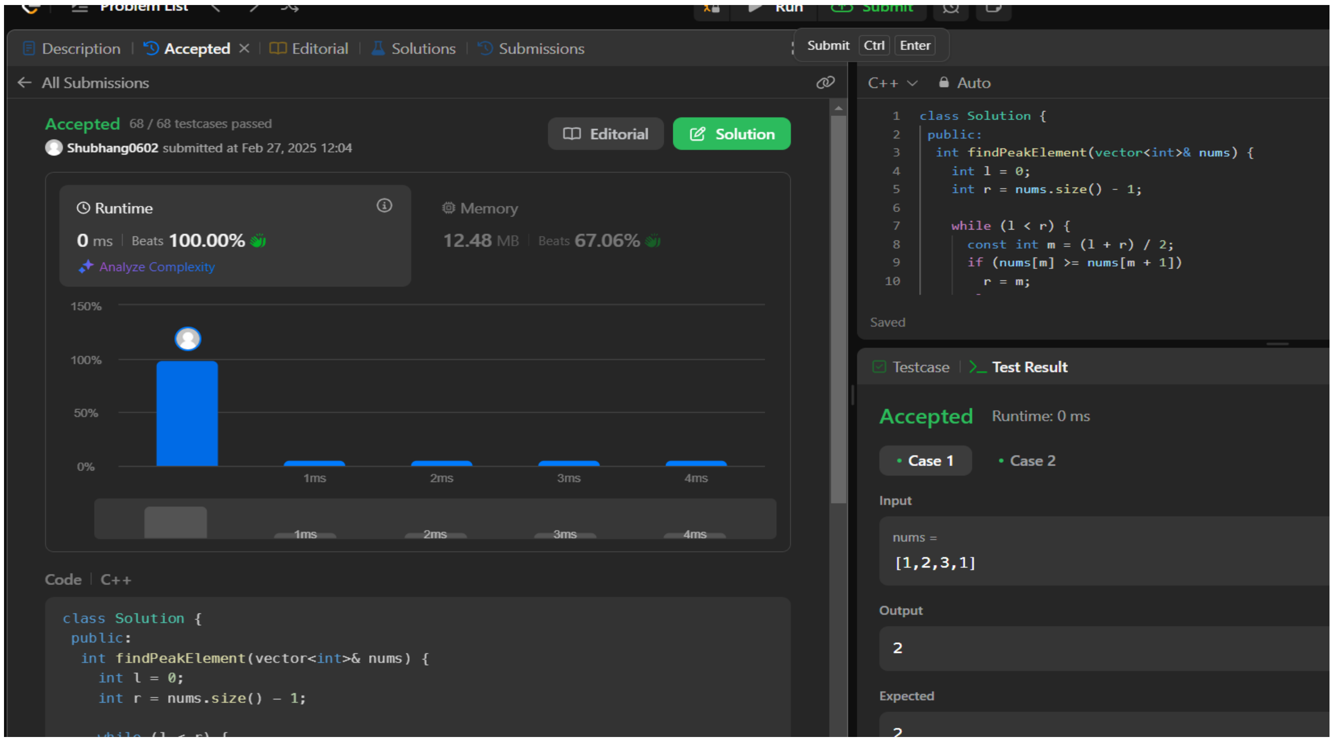
```
nums =
[1,2,3,1]
```

Output

```
2
```

Expected

```
2
```



Accepted 68 / 68 testcases passed

Shubhang0602 submitted at Feb 27, 2025 12:04

Runtime: 0 ms | Beats 100.00%

Memory: 12.48 MB | Beats 67.06%

Code | C++

```cpp
class Solution {
public:
  int findPeakElement(vector<int>& nums) {
    int l = 0;
    int r = nums.size() - 1;

    while (l < r) {
      const int m = (l + r) / 2;
      if (nums[m] >= nums[m + 1])
        r = m;
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

```
nums =
[1,2,3,1]
```

Output

```
2
```

Expected

```
2
```

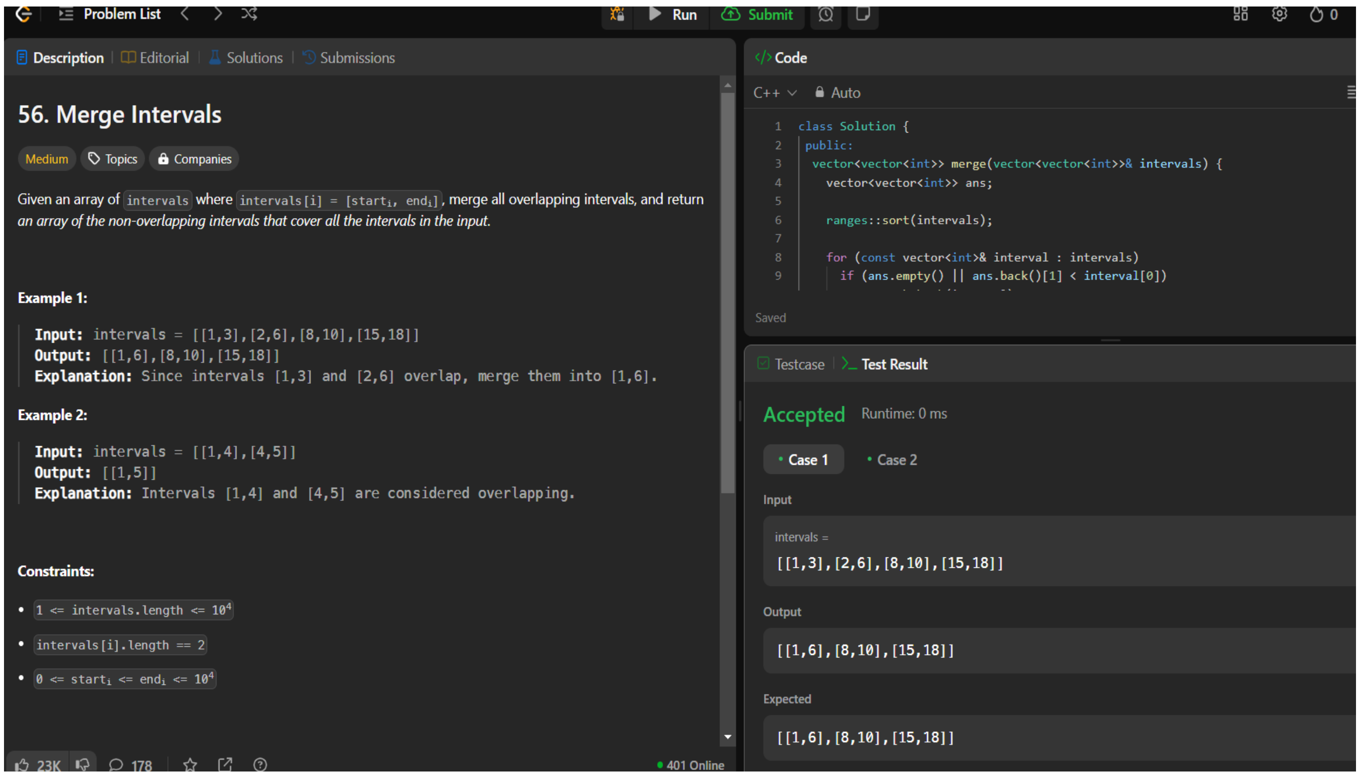
PROBLEM 7:

Aim: Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.

Code:

```
class Solution {
 public:
  vector<vector<int>> merge(vector<vector<int>>& intervals) {
    vector<vector<int>> ans;

    ranges::sort(intervals);

    for (const vector<int>& interval : intervals)
      if (ans.empty() || ans.back()[1] < interval[0])
        ans.push_back(interval);
      else
        ans.back()[1] = max(ans.back()[1], interval[1]);

    return ans;
  }
};
```

Output:



56. Merge Intervals

Medium · Topics · Companies

Given an array of `intervals` where `intervals[i] = [start_i, end_i]`, merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input*.

Example 1:

```
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].
```

Example 2:

```
Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.
```

Constraints:

- `1 <= intervals.length <= 10^4`
- `intervals[i].length == 2`
- `0 <= start_i <= end_i <= 10^4`

23K · 178 · 401 Online

```
class Solution {
public:
  vector<vector<int>> merge(vector<vector<int>>& intervals) {
    vector<vector<int>> ans;

    ranges::sort(intervals);

    for (const vector<int>& interval : intervals)
      if (ans.empty() || ans.back()[1] < interval[0])
```

Accepted Runtime: 0 ms

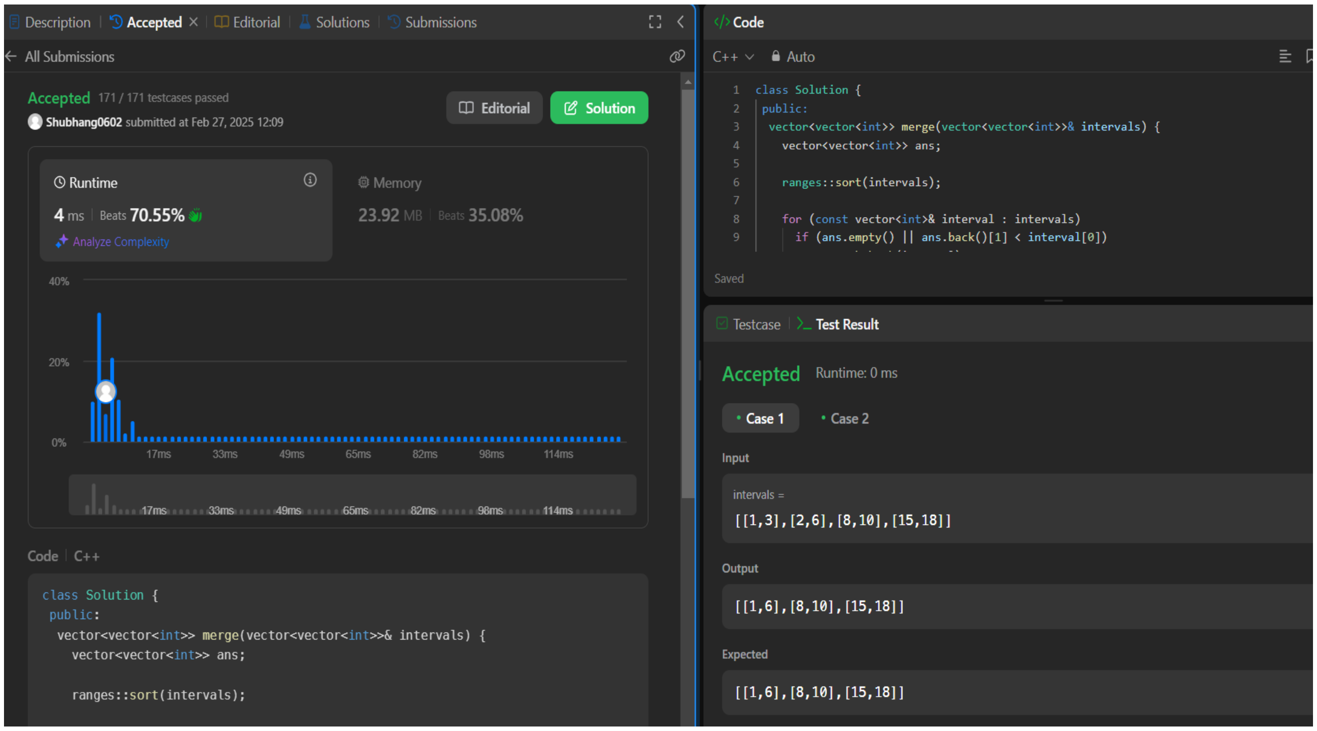
Case 1 · Case 2

Input

intervals =
```
[[1,3],[2,6],[8,10],[15,18]]
```

Output
```
[[1,6],[8,10],[15,18]]
```

Expected
```
[[1,6],[8,10],[15,18]]
```



Accepted 171 / 171 testcases passed
Shubhang0602 submitted at Feb 27, 2025 12:09

Runtime 4 ms | Beats 70.55%
Analyze Complexity
Memory 23.92 MB | Beats 35.08%

Code | C++

```
class Solution {
 public:
  vector<vector<int>> merge(vector<vector<int>>& intervals) {
    vector<vector<int>> ans;

    ranges::sort(intervals);

    for (const vector<int>& interval : intervals)
      if (ans.empty() || ans.back()[1] < interval[0])
```

Saved

Accepted Runtime: 0 ms

Input
intervals =
[[1,3],[2,6],[8,10],[15,18]]

Output
[[1,6],[8,10],[15,18]]

Expected
[[1,6],[8,10],[15,18]]

PROBLEM 8:

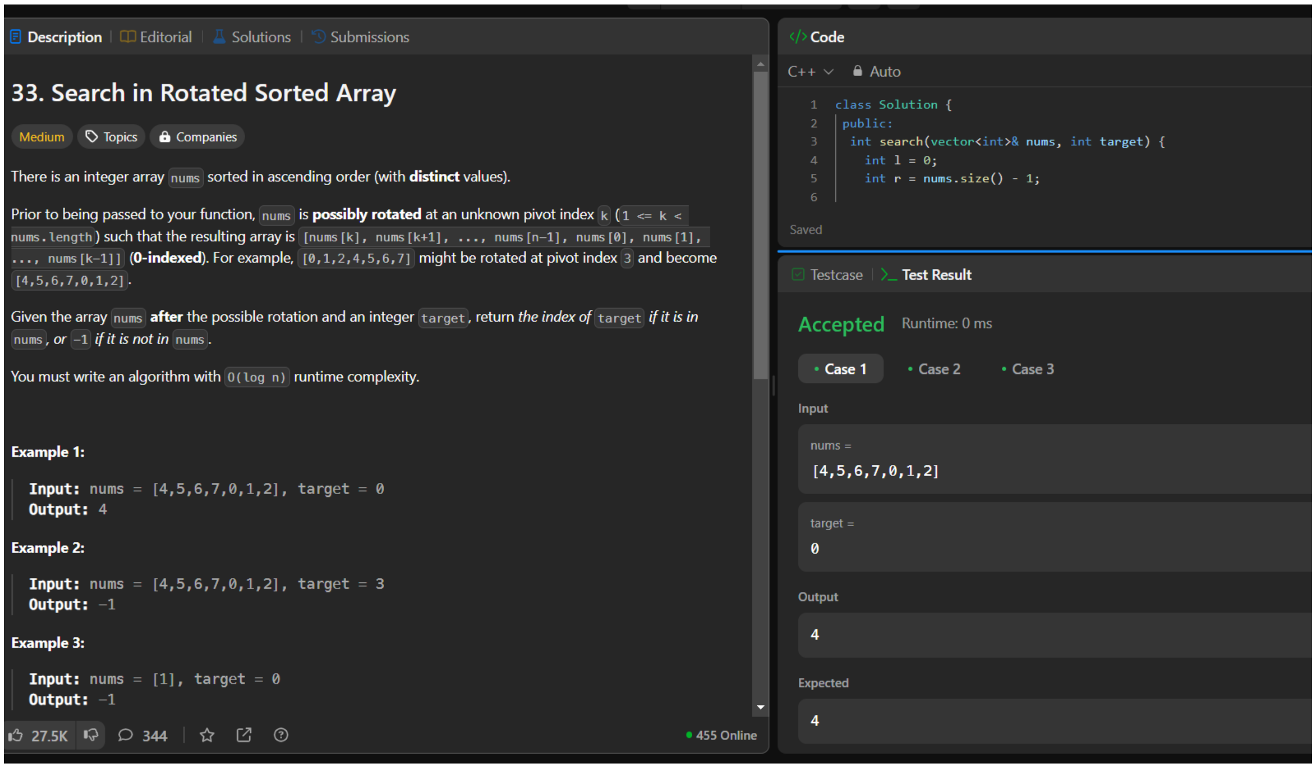
Aim: There is an integer array nums sorted in ascending order (with distinct values).
Prior to being passed to your function, nums is possibly rotated at an unknown pivot index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For example, [0,1,2,4,5,6,7] might be rotated at pivot index 3 and become [4,5,6,7,0,1,2].
Given the array nums after the possible rotation and an integer target, return the index of target if it is in nums, or -1 if it is not in nums.
You must write an algorithm with O(log n) runtime complexity.

Code:

```
class Solution {
 public:
  int search(vector<int>& nums, int target) {
    int l = 0;
    int r = nums.size() - 1;

    while (l <= r) {
      const int m = (l + r) / 2;
      if (nums[m] == target)
        return m;
      if (nums[l] <= nums[m]) {  // nums[l..m] are sorted.
        if (nums[l] <= target && target < nums[m])
```

```
          r = m - 1;
        else
          l = m + 1;
      } else {  // nums[m..n - 1] are sorted.
        if (nums[m] < target && target <= nums[r])
          l = m + 1;
        else
          r = m - 1;
      }
    }

    return -1;
  }
};
```

Output:



Description | Editorial | Solutions | Submissions

33. Search in Rotated Sorted Array

Medium | Topics | Companies

There is an integer array `nums` sorted in ascending order (with **distinct** values).

Prior to being passed to your function, `nums` is **possibly rotated** at an unknown pivot index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be rotated at pivot index `3` and become `[4,5,6,7,0,1,2]`.

Given the array `nums` **after** the possible rotation and an integer `target`, return *the index of* `target` *if it is in* `nums`, *or* `-1` *if it is not in* `nums`.

You must write an algorithm with `O(log n)` runtime complexity.

Example 1:

```
Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

Example 3:

```
Input: nums = [1], target = 0
Output: -1
```

27.5K | 344 | 455 Online

Code

C++ | Auto

```
class Solution {
public:
  int search(vector<int>& nums, int target) {
    int l = 0;
    int r = nums.size() - 1;

```

Saved

Testcase | Test Result

Accepted   Runtime: 0 ms

Case 1 | Case 2 | Case 3

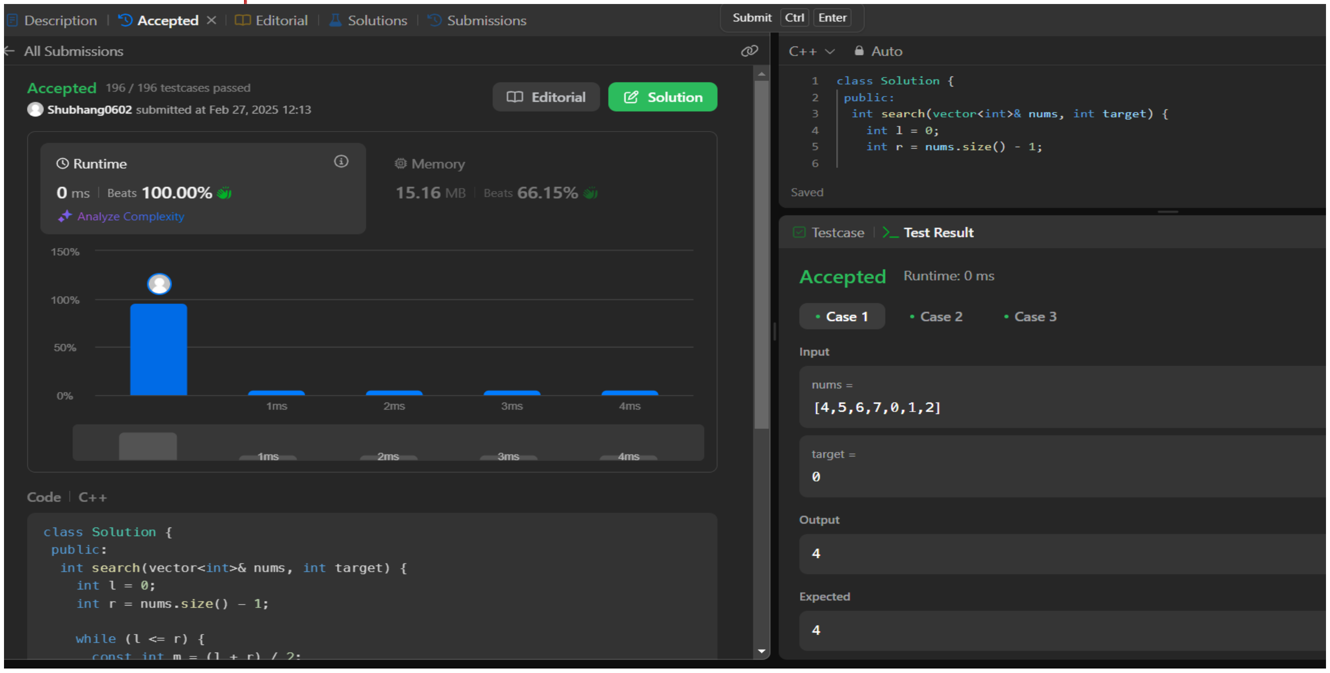
Input

nums =
```
[4,5,6,7,0,1,2]
```

target =
```
0
```

Output
```
4
```

Expected
```
4
```



Accepted 196 / 196 testcases passed
Shubhang0602 submitted at Feb 27, 2025 12:13

Runtime
0 ms | Beats 100.00%

Memory
15.16 MB | Beats 66.15%

Code | C++

```
class Solution {
 public:
  int search(vector<int>& nums, int target) {
    int l = 0;
    int r = nums.size() - 1;

    while (l <= r) {
      const int m = (l + r) / 2;
```

```
class Solution {
public:
  int search(vector<int>& nums, int target) {
    int l = 0;
    int r = nums.size() - 1;
```

Accepted Runtime: 0 ms

Input

nums =
```
[4,5,6,7,0,1,2]
```

target =
```
0
```

Output
```
4
```

Expected
```
4
```

PROBLEM 10:

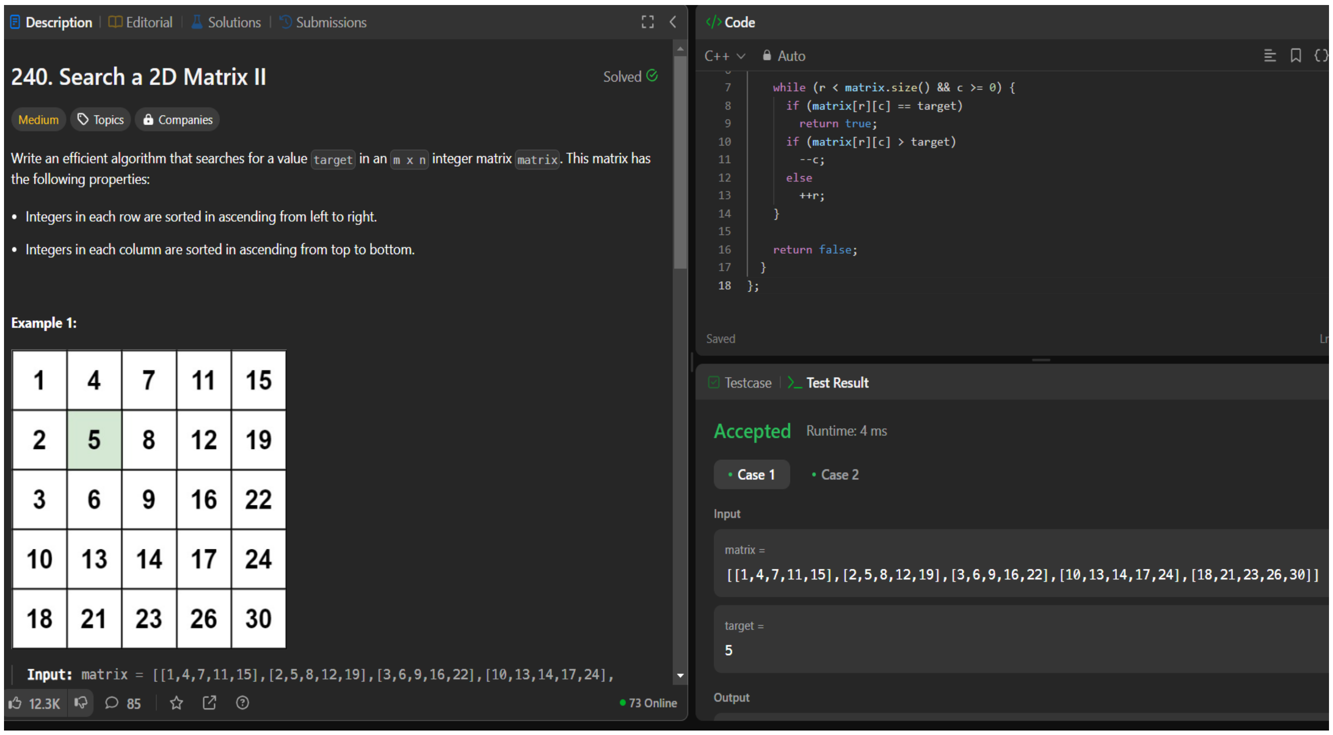
Aim: Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. This matrix has the following properties:

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

Code:

```
class Solution {
 public:
  bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int r = 0;
    int c = matrix[0].size() - 1;

    while (r < matrix.size() && c >= 0) {
      if (matrix[r][c] == target)
        return true;
      if (matrix[r][c] > target)
        --c;
      else
        ++r;
    }

    return false;
  }
};
```

Output:



240. Search a 2D Matrix II — Solved

Medium · Topics · Companies

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.

Example 1:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],

12.3K · 85 · 73 Online

```
        while (r < matrix.size() && c >= 0) {
          if (matrix[r][c] == target)
            return true;
          if (matrix[r][c] > target)
            --c;
          else
            ++r;
        }

        return false;
    }
};
```

Saved

Testcase | Test Result

Accepted Runtime: 4 ms

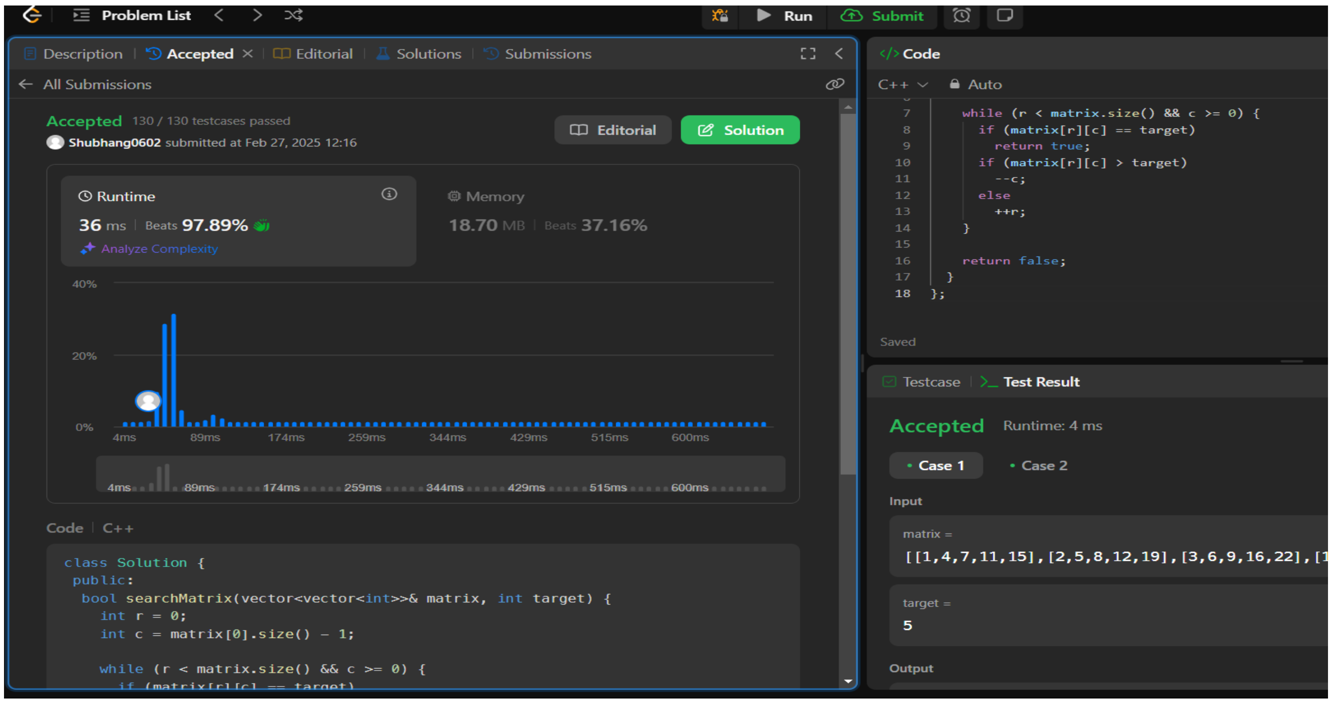
Case 1 · Case 2

Input

matrix =
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]

target =
5

Output



Accepted 130 / 130 testcases passed

Shubhang0602 submitted at Feb 27, 2025 12:16

Runtime: 36 ms | Beats 97.89%

Memory: 18.70 MB | Beats 37.16%

Code | C++

```
class Solution {
 public:
  bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int r = 0;
    int c = matrix[0].size() - 1;

    while (r < matrix.size() && c >= 0) {
      if (matrix[r][c] == target)
```

```
        while (r < matrix.size() && c >= 0) {
          if (matrix[r][c] == target)
            return true;
          if (matrix[r][c] > target)
            --c;
          else
            ++r;
        }

        return false;
    }
};
```

Saved

Testcase | Test Result

Accepted Runtime: 4 ms

Case 1 · Case 2

Input

matrix =
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[1

target =
5

Output

PROBLEM 11:

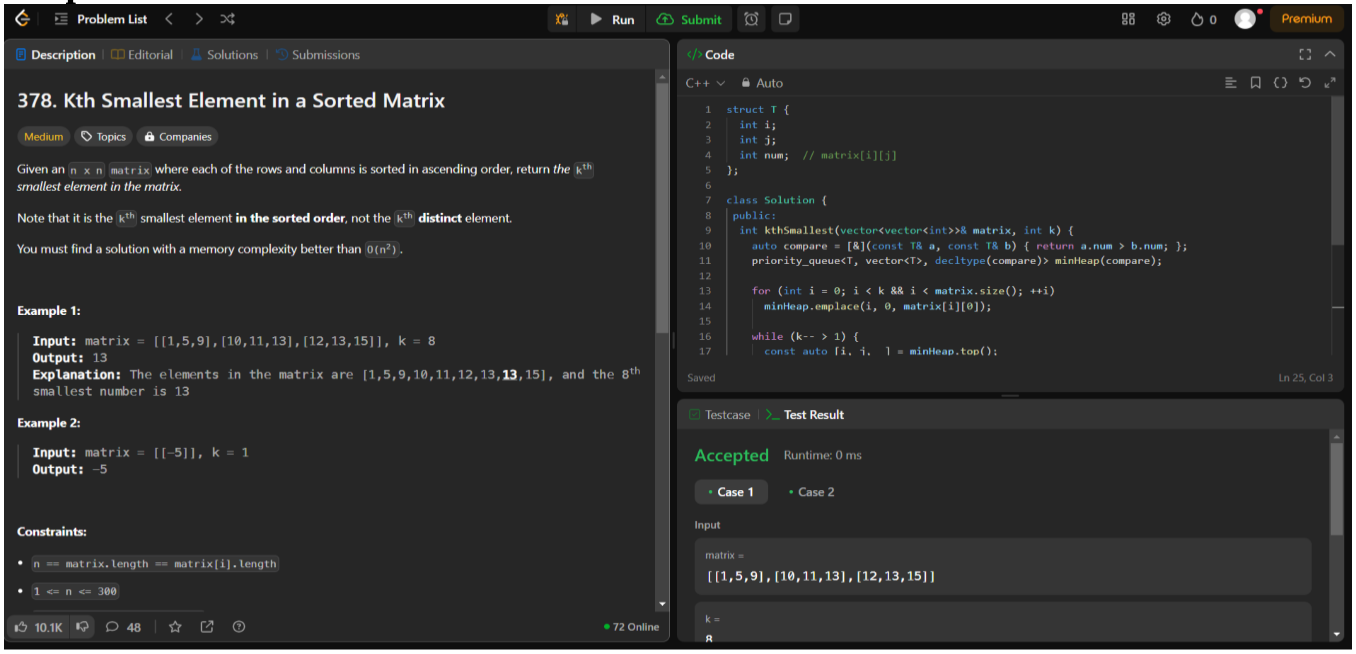
Aim: Given an n x n matrix where each of the rows and columns is sorted in ascending order, return the kth smallest element in the matrix.
Note that it is the kth smallest element in the sorted order, not the kth distinct element.

You must find a solution with a memory complexity better than O(n2).

Code:

```
struct T {
  int i;
  int j;
  int num;
};

class Solution {
 public:
  int kthSmallest(vector<vector<int>>& matrix, int k) {
    auto compare = [&](const T& a, const T& b) { return a.num > b.num; };
    priority_queue<T, vector<T>, decltype(compare)> minHeap(compare);

    for (int i = 0; i < k && i < matrix.size(); ++i)
      minHeap.emplace(i, 0, matrix[i][0]);

    while (k-- > 1) {
      const auto [i, j, _] = minHeap.top();
      minHeap.pop();
      if (j + 1 < matrix[0].size())
        minHeap.emplace(i, j + 1, matrix[i][j + 1]);
    }

    return minHeap.top().num;
  }
};
```

Output:



378. Kth Smallest Element in a Sorted Matrix

Given an `n x n` `matrix` where each of the rows and columns is sorted in ascending order, return *the* `k^th` *smallest element in the matrix.*

Note that it is the `k^th` smallest element **in the sorted order**, not the `k^th` **distinct** element.

You must find a solution with a memory complexity better than `O(n^2)`.

Example 1:

```
Input: matrix = [[1,5,9],[10,11,13],[12,13,15]], k = 8
Output: 13
Explanation: The elements in the matrix are [1,5,9,10,11,12,13,13,15], and the 8th
smallest number is 13
```

Example 2:

```
Input: matrix = [[-5]], k = 1
Output: -5
```

Constraints:

- `n == matrix.length == matrix[i].length`
- `1 <= n <= 300`

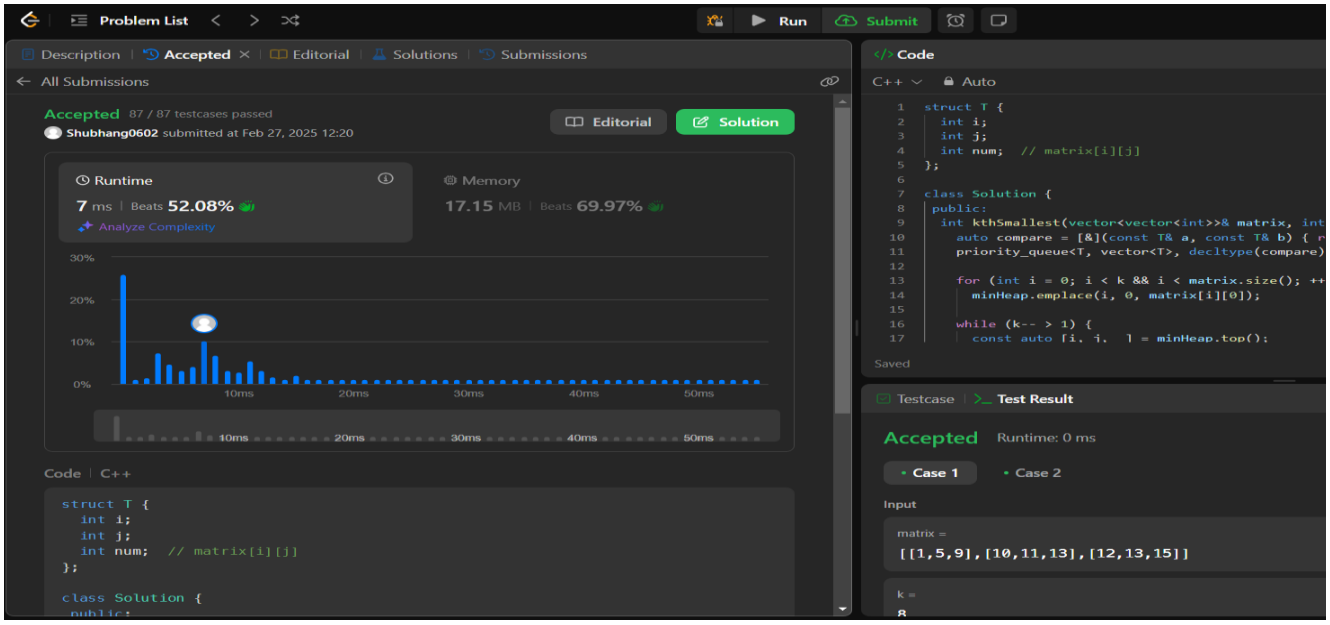
```
struct T {
  int i;
  int j;
  int num;  // matrix[i][j]
};

class Solution {
 public:
  int kthSmallest(vector<vector<int>>& matrix, int k) {
    auto compare = [&](const T& a, const T& b) { return a.num > b.num; };
    priority_queue<T, vector<T>, decltype(compare)> minHeap(compare);

    for (int i = 0; i < k && i < matrix.size(); ++i)
      minHeap.emplace(i, 0, matrix[i][0]);

    while (k-- > 1) {
      const auto [i, j, _] = minHeap.top();
```

Accepted Runtime: 0 ms

Input

matrix =
[[1,5,9],[10,11,13],[12,13,15]]

k =
8



Accepted 87 / 87 testcases passed
Shubhang0602 submitted at Feb 27, 2025 12:20

Runtime
7 ms | Beats 52.08%

Memory
17.15 MB | Beats 69.97%

Code | C++

```
struct T {
  int i;
  int j;
  int num;  // matrix[i][j]
};

class Solution {
 public:
```

Code
C++ Auto

```
struct T {
  int i;
  int j;
  int num;  // matrix[i][j]
};

class Solution {
 public:
  int kthSmallest(vector<vector<int>>& matrix, int
    auto compare = [&](const T& a, const T& b) { r
    priority_queue<T, vector<T>, decltype(compare)

    for (int i = 0; i < k && i < matrix.size(); ++
      minHeap.emplace(i, 0, matrix[i][0]);

    while (k-- > 1) {
      const auto [i, j, _] = minHeap.top();
```

Accepted Runtime: 0 ms

Input

matrix =
[[1,5,9],[10,11,13],[12,13,15]]

k =
8

PROBLEM 12:

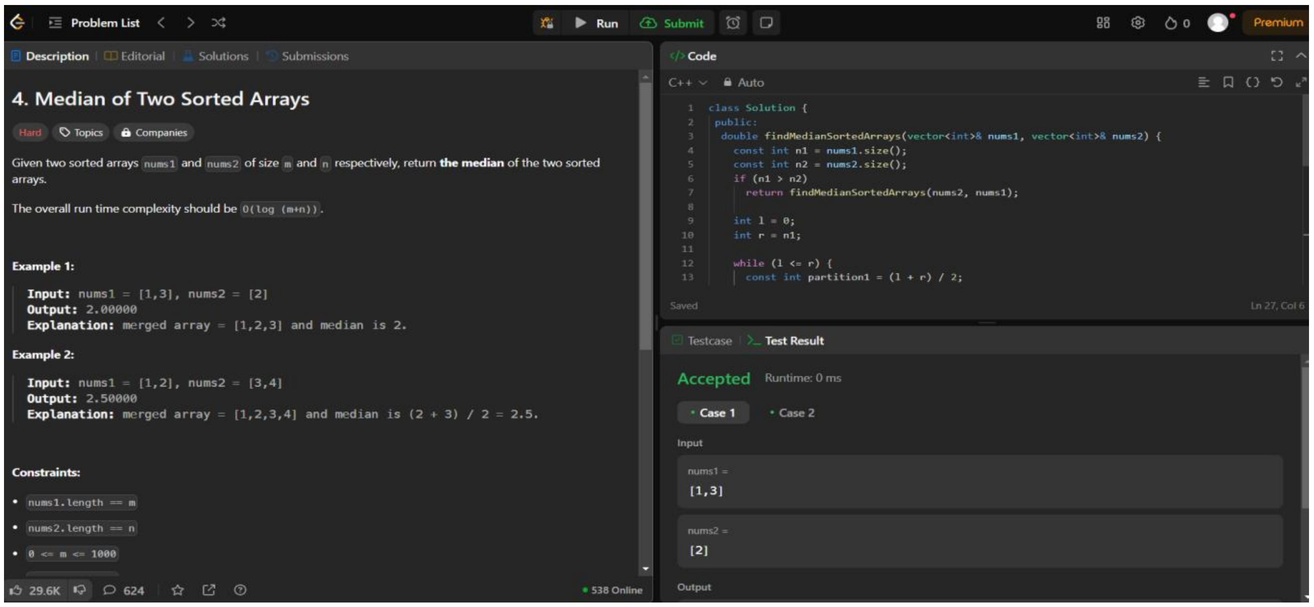
Aim: Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays.
The overall run time complexity should be O(log (m+n)).

Code:

```
class Solution {
 public:
  double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    const int n1 = nums1.size();
    const int n2 = nums2.size();
    if (n1 > n2)
      return findMedianSortedArrays(nums2, nums1);

    int l = 0;
    int r = n1;

    while (l <= r) {
      const int partition1 = (l + r) / 2;
      const int partition2 = (n1 + n2 + 1) / 2 - partition1;
      const int maxLeft1 = partition1 == 0 ? INT_MIN : nums1[partition1 - 1];
      const int maxLeft2 = partition2 == 0 ? INT_MIN : nums2[partition2 - 1];
      const int minRight1 = partition1 == n1 ? INT_MAX : nums1[partition1];
      const int minRight2 = partition2 == n2 ? INT_MAX : nums2[partition2];
      if (maxLeft1 <= minRight2 && maxLeft2 <= minRight1)
        return (n1 + n2) % 2 == 0
                   ? (max(maxLeft1, maxLeft2) + min(minRight1, minRight2)) * 0.5
```

```
                    : max(maxLeft1, maxLeft2);
      else if (maxLeft1 > minRight2)
        r = partition1 - 1;
      else
        l = partition1 + 1;
    }

    throw;
  }
};
```

Output:



4. Median of Two Sorted Arrays

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be `O(log (m+n))`.

Example 1:

```
Input: nums1 = [1,3], nums2 = [2]
Output: 2.00000
Explanation: merged array = [1,2,3] and median is 2.
```

Example 2:

```
Input: nums1 = [1,2], nums2 = [3,4]
Output: 2.50000
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.
```

Constraints:

- `nums1.length == m`
- `nums2.length == n`
- `0 <= m <= 1000`

```
class Solution {
 public:
  double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    const int n1 = nums1.size();
    const int n2 = nums2.size();
    if (n1 > n2)
      return findMedianSortedArrays(nums2, nums1);

    int l = 0;
    int r = n1;

    while (l <= r) {
      const int partition1 = (l + r) / 2;
```

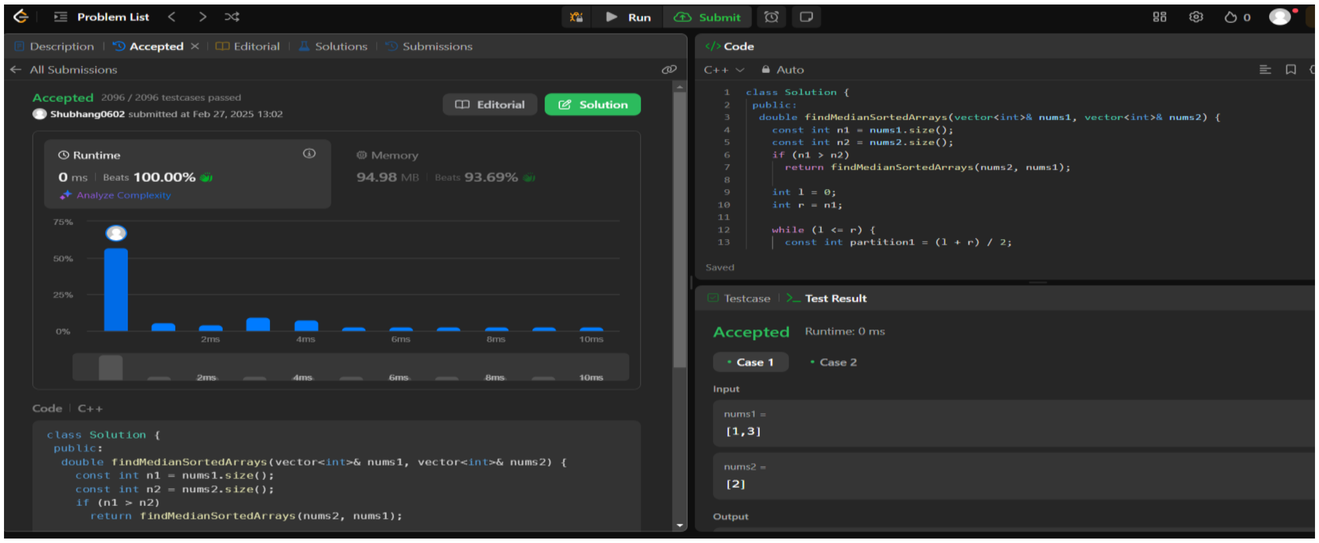
Accepted Runtime: 0 ms

Input

nums1 =
[1,3]

nums2 =
[2]

Output



Accepted 2096 / 2096 testcases passed
Shubhang0602 submitted at Feb 27, 2025 13:02

Runtime
0 ms | Beats 100.00%

Memory
94.98 MB | Beats 93.69%

Code | C++

```
class Solution {
 public:
  double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    const int n1 = nums1.size();
    const int n2 = nums2.size();
    if (n1 > n2)
      return findMedianSortedArrays(nums2, nums1);
```

```
class Solution {
 public:
  double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    const int n1 = nums1.size();
    const int n2 = nums2.size();
    if (n1 > n2)
      return findMedianSortedArrays(nums2, nums1);

    int l = 0;
    int r = n1;

    while (l <= r) {
      const int partition1 = (l + r) / 2;
```

Accepted Runtime: 0 ms

Input

nums1 =
[1,3]

nums2 =
[2]

Output