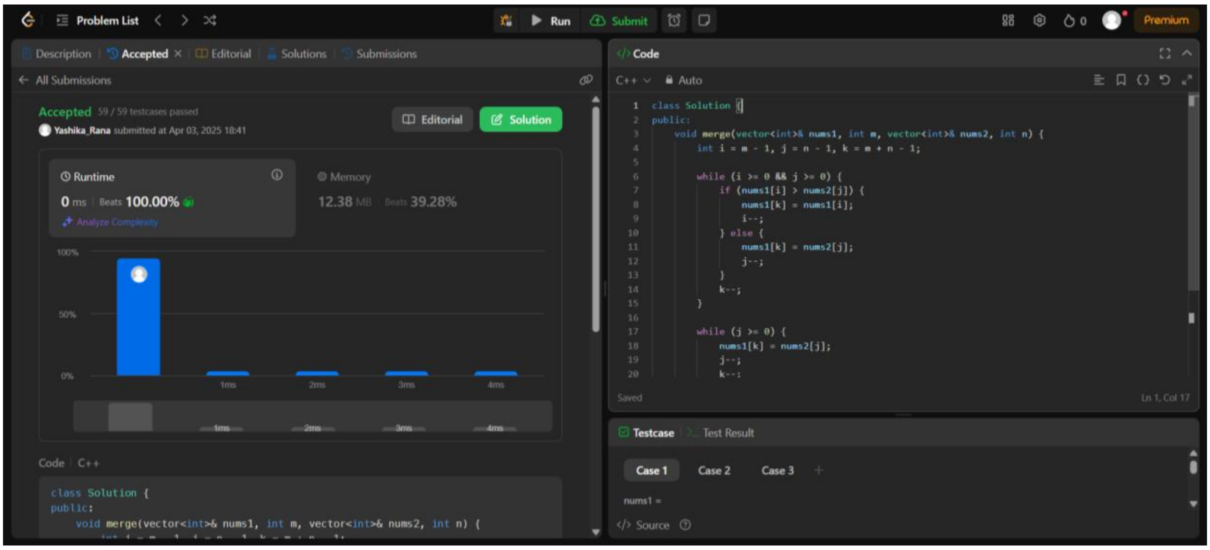
YASHIKA **22BCS10295**

ASSIGNMENT- 5

Problem 1: Merge Sorted Array

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

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

```
        while (j >= 0)
    {
        nums1[k] = nums2[j];
            j--;
            k--;
        }
      }
  };
```



Accepted 59 / 59 testcases passed

Yashika_Rana submitted at Apr 03, 2025 18:41

Runtime
0 ms | Beats 100.00%

Memory
12.38 MB | Beats 39.28%

Code | C++

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
```

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

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

        while (j >= 0) {
            nums1[k] = nums2[j];
            j--;
            k--;
```

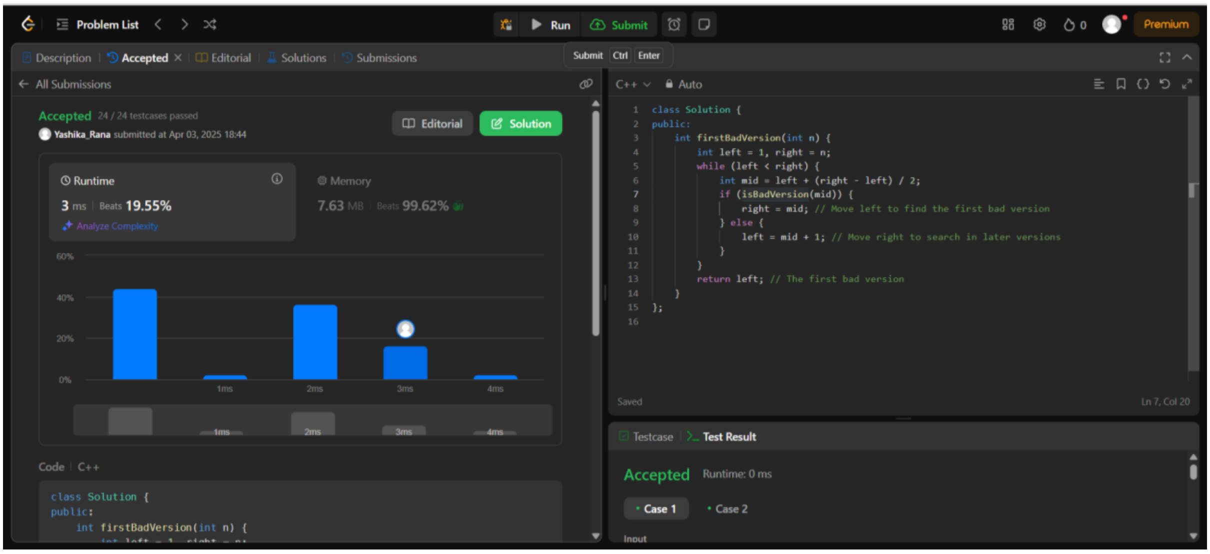
Saved

Ln 1, Col 17

nums1 =

Problem 2: First Bad Version

```
class Solution {
public:
    int firstBadVersion(int n) {
        int left = 1, right = n;
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (isBadVersion(mid)) {
                right = mid; // Move left to find the first bad version
            } else {
                left = mid + 1; // Move right to search in later versions
            }
        }
        return left; // The first bad version
    }
};
```



Accepted 24 / 24 testcases passed

Yashika_Rana submitted at Apr 03, 2025 18:44

Runtime: 3 ms | Beats 19.55%

Memory: 7.63 MB | Beats 99.62%

Code | C++

```
class Solution {
public:
    int firstBadVersion(int n) {
        int left = 1, right = n;
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (isBadVersion(mid)) {
                right = mid; // Move left to find the first bad version
            } else {
                left = mid + 1; // Move right to search in later versions
            }
        }
        return left; // The first bad version
    }
};
```

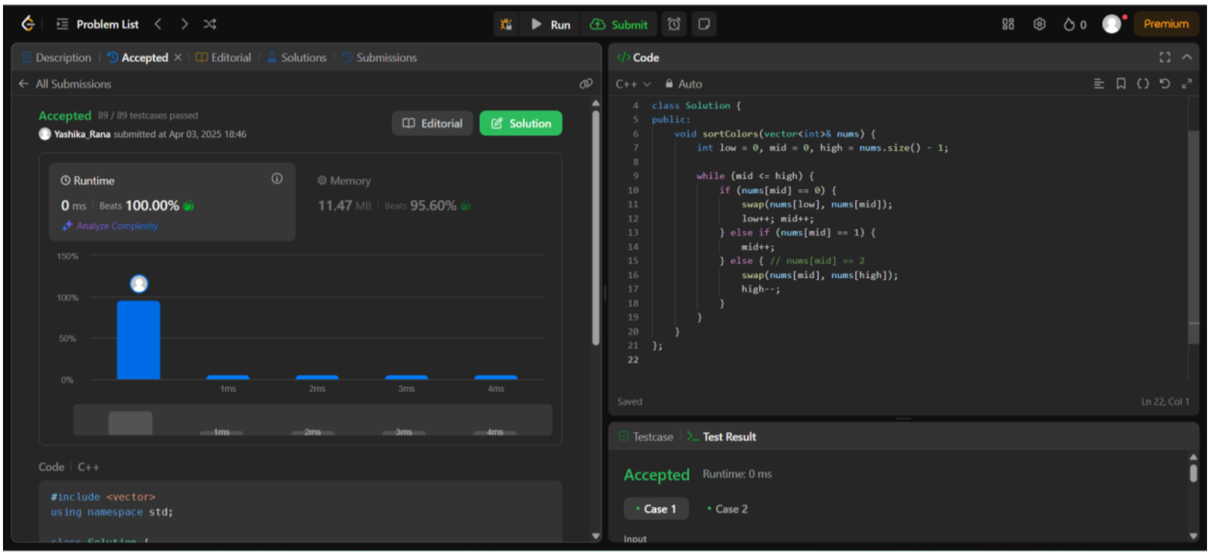
Saved

Ln 7, Col 20

Accepted Runtime: 0 ms

Problem 3: Sort Colors

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low = 0, mid = 0, high = nums.size() - 1;
        while (mid <= high) {
            if (nums[mid] == 0) {
                swap(nums[low], nums[mid]);
                low++; mid++;
            } else if (nums[mid] == 1) {
                mid++;
            } else { // nums[mid] == 2
                swap(nums[mid], nums[high]);
                high--;
            }  }
    }
};
```



Accepted 89 / 89 testcases passed

Yashika_Rana submitted at Apr 03, 2025 18:46

Runtime 0 ms | Beats 100.00%

Memory 11.47 MB | Beats 95.60%

Code | C++

```
#include <vector>
using namespace std;

class Solution {
```

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low = 0, mid = 0, high = nums.size() - 1;

        while (mid <= high) {
            if (nums[mid] == 0) {
                swap(nums[low], nums[mid]);
                low++; mid++;
            } else if (nums[mid] == 1) {
                mid++;
            } else { // nums[mid] == 2
                swap(nums[mid], nums[high]);
                high--;
            }
        }
    }
};
```

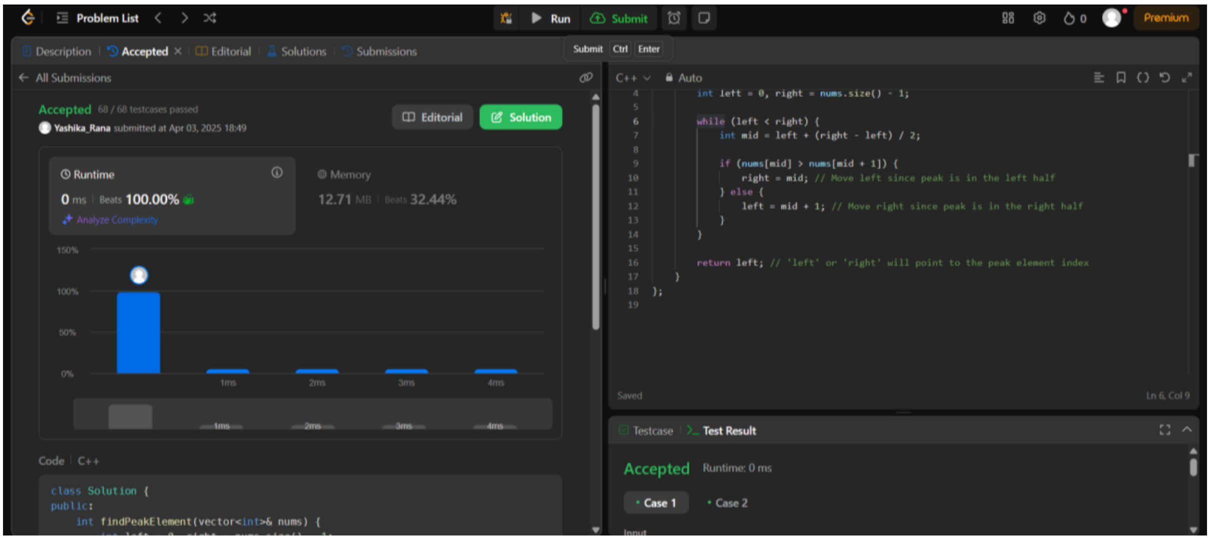
Saved

Ln 22, Col 1

Accepted Runtime: 0 ms

Problem 4: Find Peak Element

```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        int left = 0, right = nums.size() - 1;
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] > nums[mid + 1]) {
                right = mid; // Move left since peak is in the left half
            } else {
                left = mid + 1; // Move right since peak is in the right half
            }
        }
        return left; // 'left' or 'right' will point to the peak element index
    }
};
```



Accepted 68 / 68 testcases passed

Yashika_Rana submitted at Apr 03, 2025 18:49

0 ms | Beats 100.00%

12.71 MB | Beats 32.44%

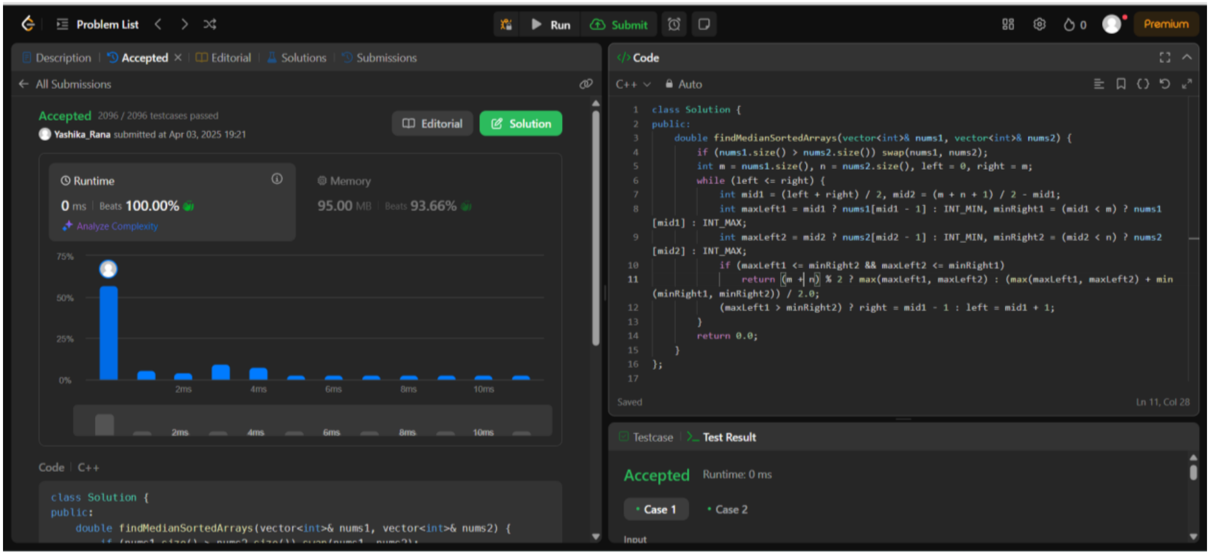
```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
```

```
        int left = 0, right = nums.size() - 1;

        while (left < right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] > nums[mid + 1]) {
                right = mid; // Move left since peak is in the left half
            } else {
                left = mid + 1; // Move right since peak is in the right half
            }
        }

        return left; // 'left' or 'right' will point to the peak element index
    }
};
```

Accepted Runtime: 0 ms

Problem 5: Median of Two Sorted Arrays

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        if (nums1.size() > nums2.size()) swap(nums1, nums2);
        int m = nums1.size(), n = nums2.size(), left = 0, right = m;
        while (left <= right) {
            int mid1 = (left + right) / 2, mid2 = (m + n + 1) / 2 - mid1;
            int maxLeft1 = mid1 ? nums1[mid1 - 1] : INT_MIN, minRight1 = (mid1 < m) ? nums1[mid1] :
            INT_MAX;
            int maxLeft2 = mid2 ? nums2[mid2 - 1] : INT_MIN, minRight2 = (mid2 < n) ? nums2[mid2] :
             INT_MAX;
            if (maxLeft1 <= minRight2 && maxLeft2 <= minRight1)
            return (m + n) % 2 ? max(maxLeft1, maxLeft2) : (max(maxLeft1, maxLeft2) + min(minRight1,
            minRight2)) / 2.0;
            (maxLeft1 > minRight2) ? right = mid1 - 1 : left = mid1 + 1;   }
        return 0.0;
    }
};
```



Accepted 2096 / 2096 testcases passed

Yashika_Rana submitted at Apr 03, 2025 19:21

Runtime

0 ms | Beats 100.00%

Memory

95.00 MB | Beats 93.66%

Code | C++

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
```

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        if (nums1.size() > nums2.size()) swap(nums1, nums2);
        int m = nums1.size(), n = nums2.size(), left = 0, right = m;
        while (left <= right) {
            int mid1 = (left + right) / 2, mid2 = (m + n + 1) / 2 - mid1;
            int maxLeft1 = mid1 ? nums1[mid1 - 1] : INT_MIN, minRight1 = (mid1 < m) ? nums1
[mid1] : INT_MAX;
            int maxLeft2 = mid2 ? nums2[mid2 - 1] : INT_MIN, minRight2 = (mid2 < n) ? nums2
[mid2] : INT_MAX;
            if (maxLeft1 <= minRight2 && maxLeft2 <= minRight1)
                return (m + n) % 2 ? max(maxLeft1, maxLeft2) : (max(maxLeft1, maxLeft2) + min
(minRight1, minRight2)) / 2.0;
            (maxLeft1 > minRight2) ? right = mid1 - 1 : left = mid1 + 1;
        }
        return 0.0;
    }
};
```

Saved

Ln 11, Col 28

Accepted Runtime: 0 ms