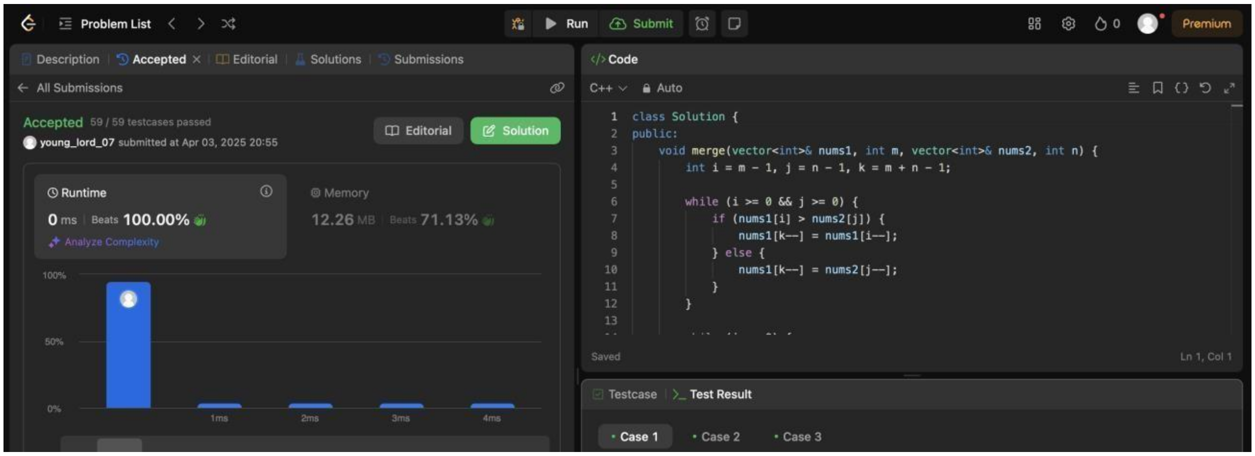
ASSIGNMENT - 5

Name: Siddhraj Gupta **UID: 22BCS15225**

Section: 608- B **Subject: AP**

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

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

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



Accepted 59 / 59 testcases passed
young_lord_07 submitted at Apr 03, 2025 20:55

Runtime
0 ms | Beats 100.00%

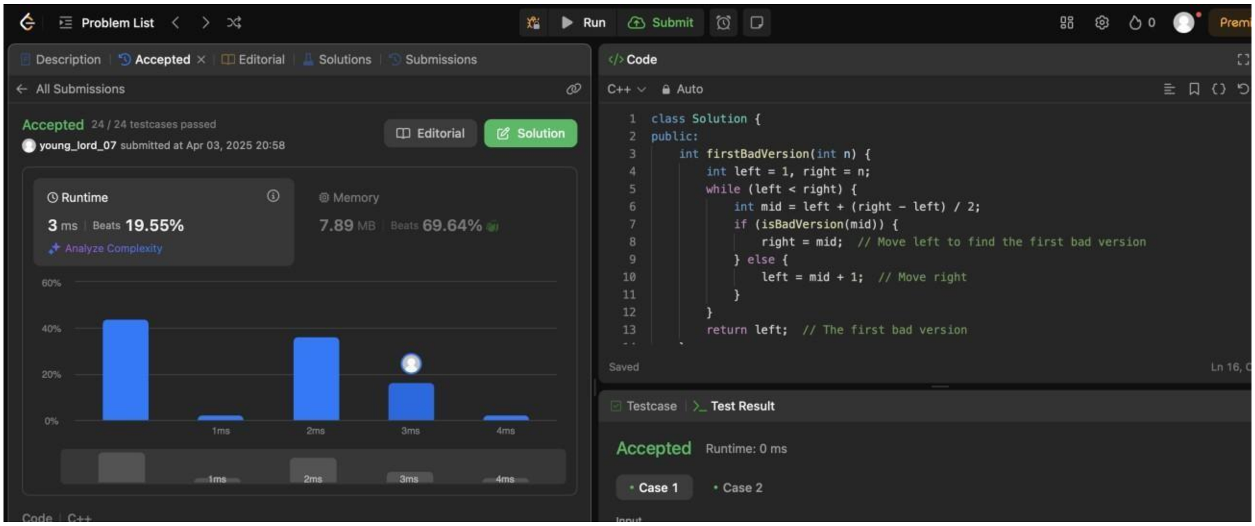
Memory
12.26 MB | Beats 71.13%

```cpp
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--];
            } else {
                nums1[k--] = nums2[j--];
            }
        }

```

Saved   Ln 1, Col 1

Solution 2: 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
      }
    }
    return left; // The first bad version
  }
};
```



Accepted 24 / 24 testcases passed

young_lord_07 submitted at Apr 03, 2025 20:58

Runtime 3 ms | Beats 19.55%

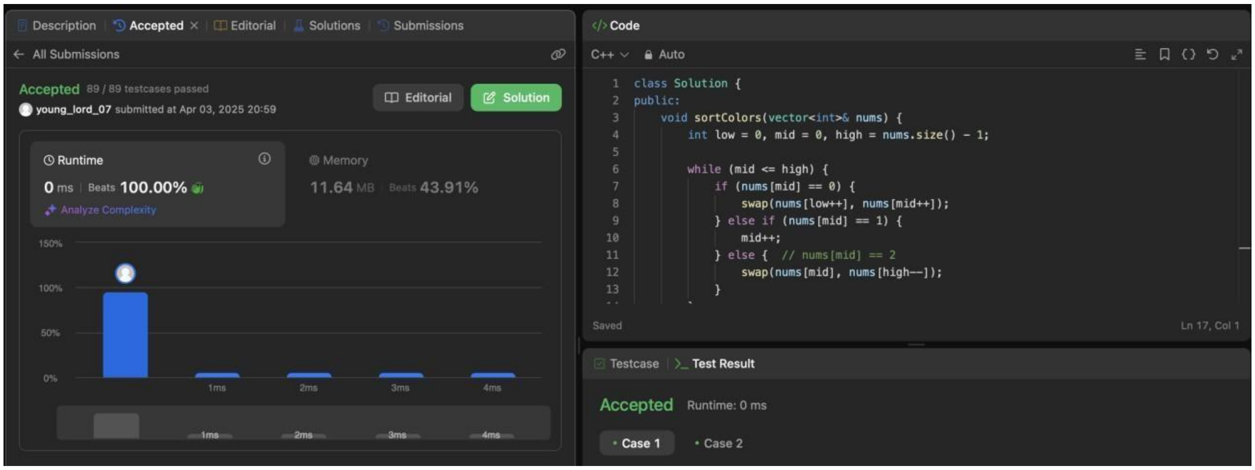
Memory 7.89 MB | Beats 69.64%

```cpp
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
            }
        }
        return left;  // The first bad version
```

Accepted Runtime: 0 ms

Solution 3: class Solution {

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


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



Accepted 89 / 89 testcases passed

young_lord_07 submitted at Apr 03, 2025 20:59

0 ms | Beats 100.00%

11.64 MB | Beats 43.91%

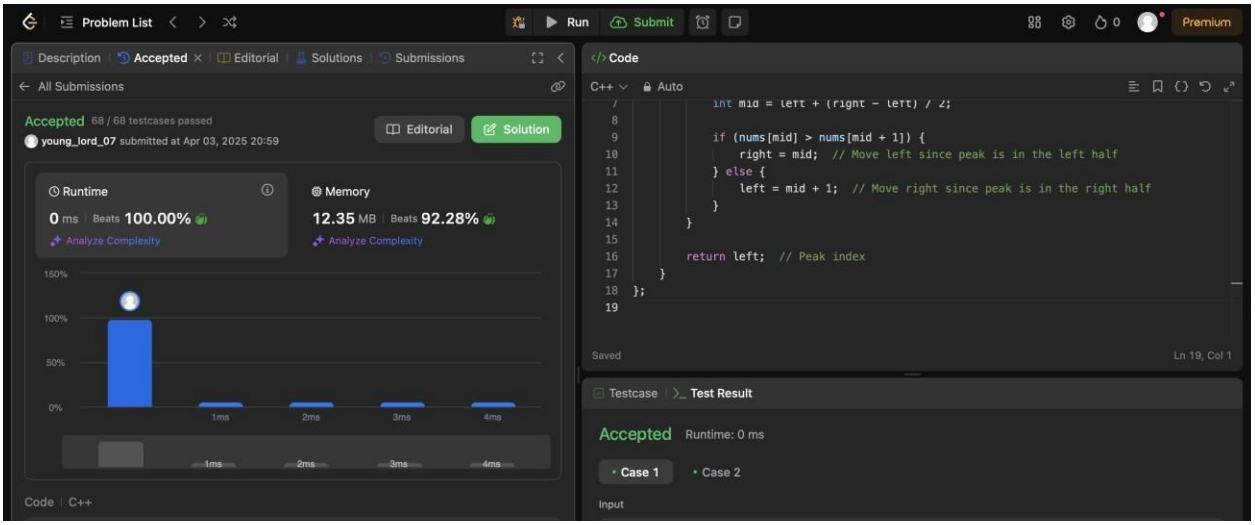
```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low = 0, mid = 0, high = nums.size() - 1;

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

Saved   Ln 17, Col 1

Accepted Runtime: 0 ms

Solution 4:

```
class Solution {
public:
  int findPeakElement(vector<int>C 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; // Peak index
  }
};
```



Accepted 68 / 68 testcases passed

young_lord_07 submitted at Apr 03, 2025 20:59

Runtime: 0 ms | Beats 100.00%

Memory: 12.35 MB | Beats 92.28%

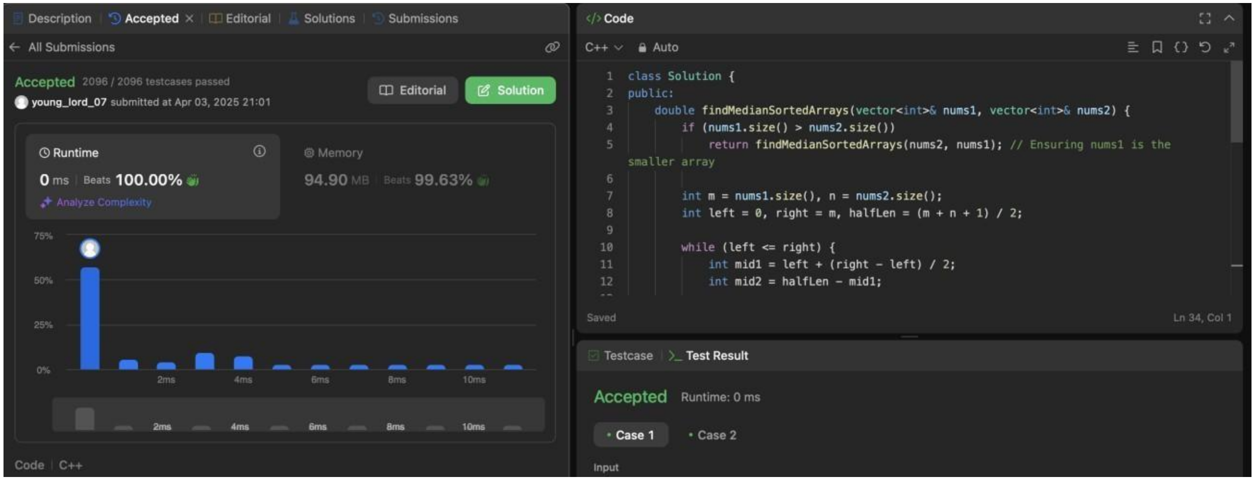
```
            int mid = left + (right - left) / 2;

            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;  // Peak index
    }
};
```

Saved — Ln 19, Col 1

Accepted Runtime: 0 ms

Solution 5:

```
class Solution { public:
   double findMedianSortedArrays(vector<int>C nums1, vector<int>C nums2)
      { if (nums1.size() > nums2.size())
  return findMedianSortedArrays(nums2, nums1); // Ensuring nums1 is the smaller
array


      int m = nums1.size(), n = nums2.size();
      int left = 0, right = m, halfLen = (m + n + 1) / 2;

 while (left <= right) { int mid1 =
      left + (right - left) / 2;
         int mid2 = halfLen - mid1;


         int left1 = (mid1 > 0) ? nums1[mid1 - 1] :
         INT_MIN; int right1 = (mid1 < m) ? nums1[mid1] :
         INT_MAX; int left2 = (mid2 > 0) ? nums2[mid2 - 1]
      : INT_MIN; int right2 = (mid2 < n) ? nums2[mid2] :
      INT_MAX;

  if (left1 <= right2 CC left2 <= right1) { if ((m + n) % 2 ==
      0) return (max(left1, left2) + min(right1, right2)) /
      2.0; else return max(left1, left2);
      } else if (left1 > right2) {
        right = mid1 - 1;
      } else { left =
        mid1 + 1;
      }
    }


    return 0.0; // This should never be reached
  }
};
```



Accepted 2096 / 2096 testcases passed

young_lord_07 submitted at Apr 03, 2025 21:01

Runtime
0 ms | Beats 100.00%

Memory
94.90 MB | Beats 99.63%

Code | C++

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        if (nums1.size() > nums2.size())
            return findMedianSortedArrays(nums2, nums1); // Ensuring nums1 is the
smaller array

        int m = nums1.size(), n = nums2.size();
        int left = 0, right = m, halfLen = (m + n + 1) / 2;

        while (left <= right) {
            int mid1 = left + (right - left) / 2;
            int mid2 = halfLen - mid1;
```

Ln 34, Col 1

Accepted Runtime: 0 ms