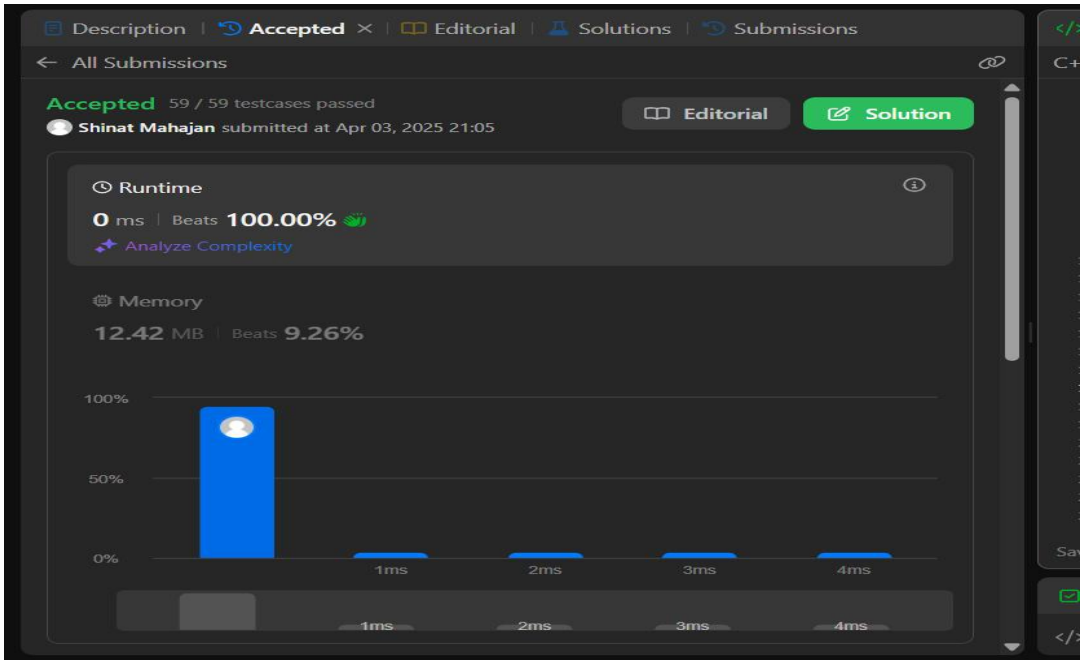
ASSIGNMENT 5

88. Merge Sorted Array

Code Snippet :

```
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 (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--;
        }
    }
};
```

Submission

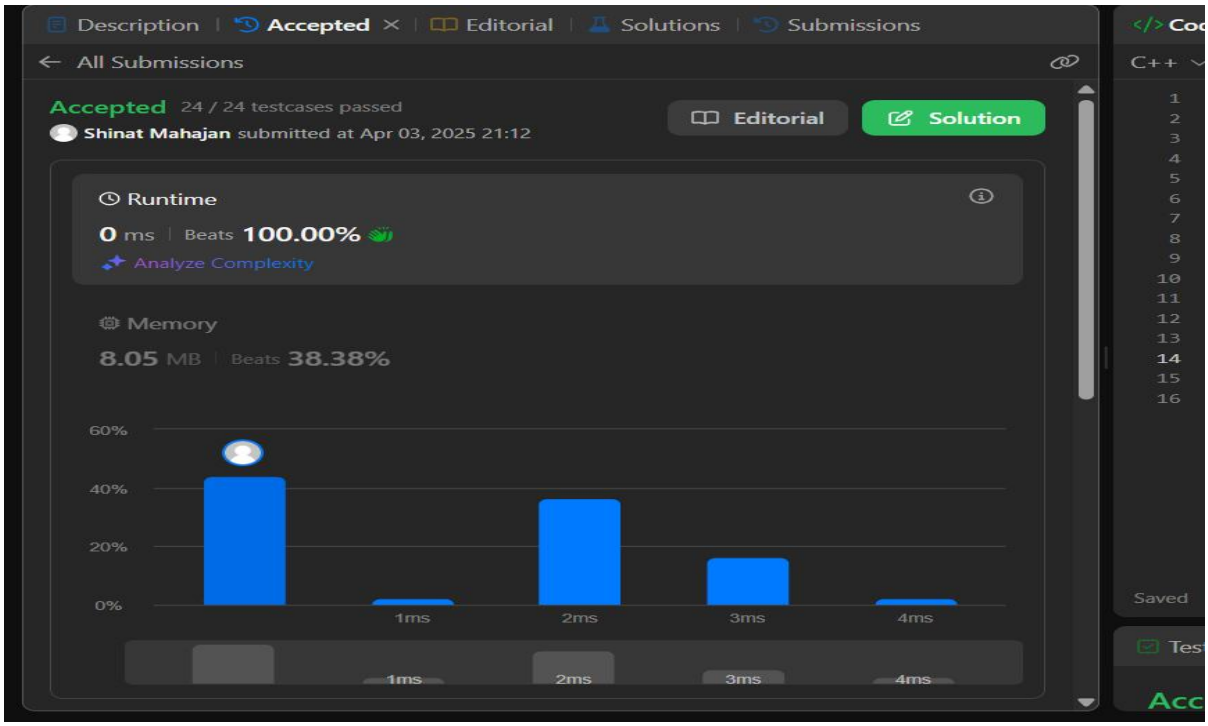

Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 59 / 59 testcases passed
Shinat Mahajan submitted at Apr 03, 2025 21:05
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
12.42 MB | Beats 9.26%
100%
50%
0%
1ms
2ms
3ms
4ms

278. First Bad Version

Code snippet

```
class Solution {
public:
    int firstBadVersion(int n) {
        int low = 1, high = n;
        while (low < high) {
            int mid = low + (high - low) / 2;
            if (isBadVersion(mid)) {
                high = mid;
            } else {
                low = mid + 1;
            }
        }
        return low;
    }
};
```

Submission

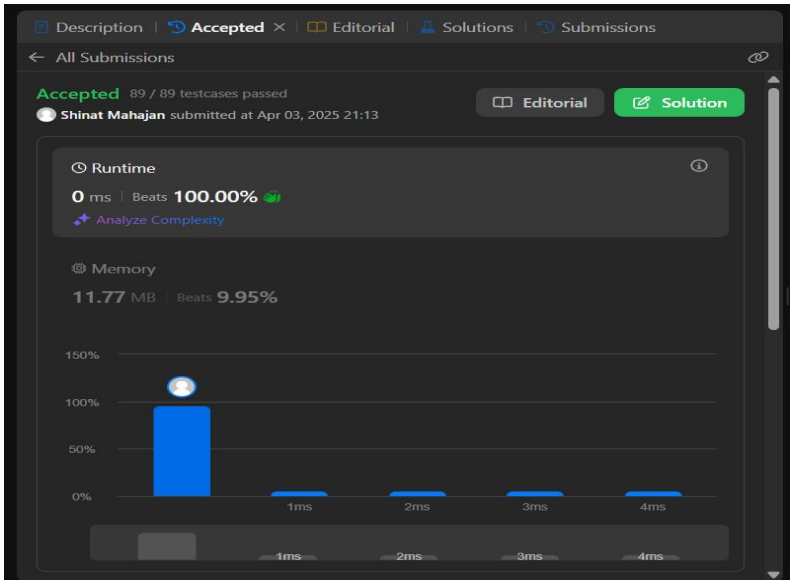


75. Sort Colors

Code snippet

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

Submission

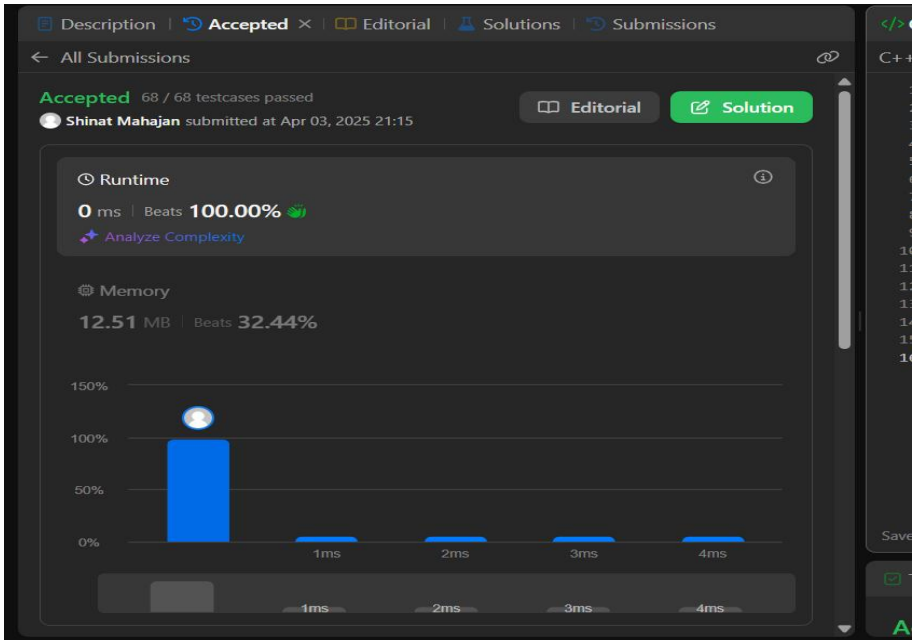


162. Find peak element

Code snippet

```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        int l =0; int r= nums.size() -1;
        while(l<r){
            int mid = l+ (r-l)/2;
            if (nums[mid]< nums[mid+1]){
                l= mid+1;
            }
            else{
                r = mid;
            }
        }
        return l;
    }
};
```

Submission



4. Median of Two Sorted Arrays

Code snippet

```
#include <vector>
#include <algorithm>
#include <limits>
using namespace std;
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        if (nums1.size() > nums2.size()) {
            return findMedianSortedArrays(nums2, nums1);
        }
        int m = nums1.size();
        int n = nums2.size();
        int totalLeft = (m + n + 1) / 2;
        int left = 0, right = m;
        while (left <= right) {
            int partition1 = (left + right) / 2;
            int partition2 = totalLeft - partition1;
            int maxLeft1 = (partition1 == 0) ? INT_MIN : nums1[partition1 - 1];
            int minRight1 = (partition1 == m) ? INT_MAX : nums1[partition1];
            int maxLeft2 = (partition2 == 0) ? INT_MIN : nums2[partition2 - 1];
            int minRight2 = (partition2 == n) ? INT_MAX : nums2[partition2];
            if (maxLeft1 <= minRight2 && maxLeft2 <= minRight1) {
                if ((m + n) % 2 == 1) {
                    return max(maxLeft1, maxLeft2);
                }
                return (max(maxLeft1, maxLeft2) + min(minRight1, minRight2)) / 2.0;
            }
            else if (maxLeft1 > minRight2) {
                right = partition1 - 1;
            }
```

```
            else {
                left = partition1 + 1;
            }
        }       throw invalid_argument("Input arrays are not sorted");
    }
};
```

Submission