Experiment 5

Student Name: Ansh **UID: 22BCS13469**
Branch: CSE **Section/Group: 608-B**
Semester: 6th **Date of Performance:13/3/25**
Subject Name: Advanced Programming - 2 **Subject Code: 22CSH-351**

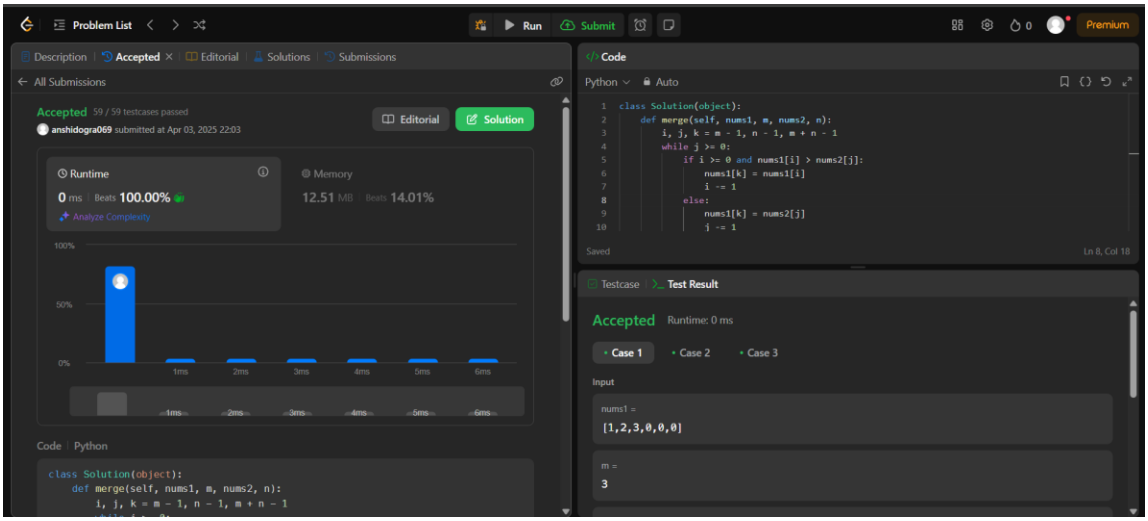
Ques 1:

Aim: Merge Sorted Array:

Code:

```
class Solution(object):
    def merge(self, nums1, m, nums2, n):
        i, j, k = m - 1, n - 1, m + n - 1
        while j >= 0:
            if i >= 0 and nums1[i] > nums2[j]:
                nums1[k] = nums1[i]
                i -= 1
            else:
                nums1[k] = nums2[j]
                j -= 1
            k -= 1
```

Submission Screenshot:



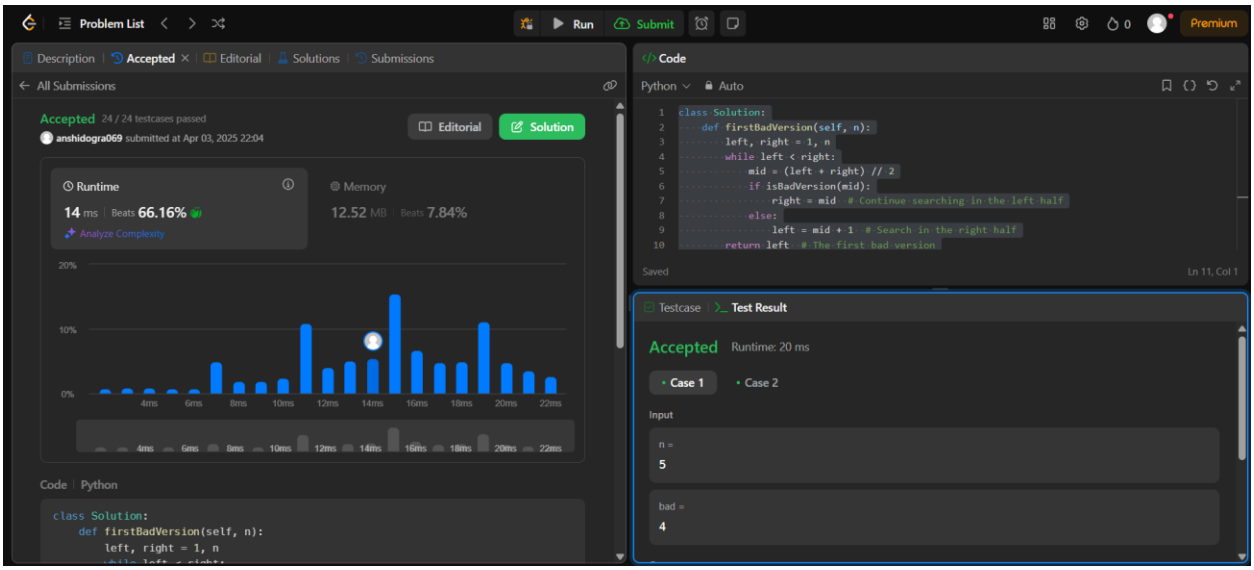
Ques 2:

Aim: First Bad Version:

Code:

```
class Solution:
    def firstBadVersion(self, n):
        left, right = 1, n
        while left < right:
            mid = (left + right) // 2
            if isBadVersion(mid):
                right = mid
            else:
                left = mid + 1
        return left
```

Submission Screenshot:



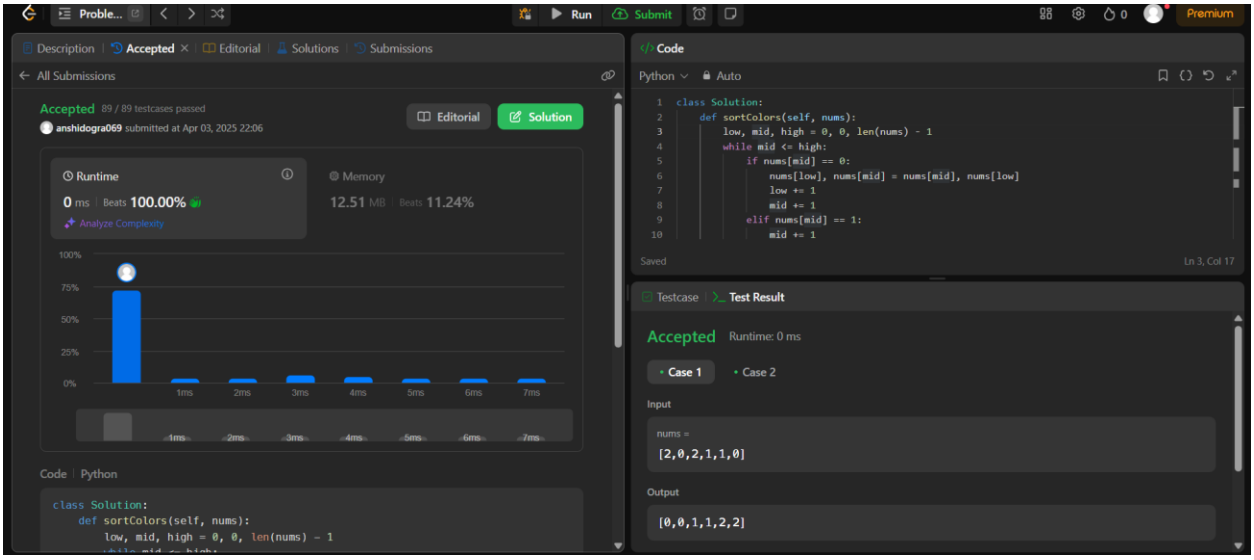
Ques 3:

Aim: Sort Colors:

Code:

```
class Solution:
    def sortColors(self, nums):
        low, mid, high = 0, 0, len(nums) - 1
        while mid <= high:
            if nums[mid] == 0:
                nums[low], nums[mid] = nums[mid], nums[low]
                low += 1
                mid += 1
            elif nums[mid] == 1:
                mid += 1
            else:
                nums[mid], nums[high] = nums[high], nums[mid]
                high -= 1
```

Submission Screenshot:



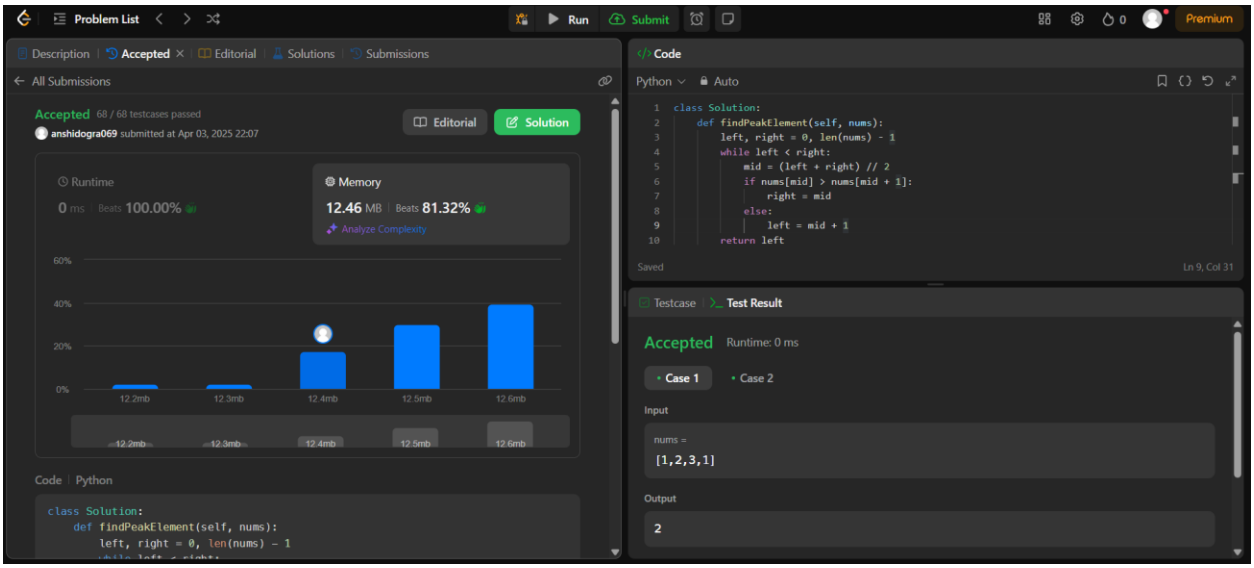
Ques 4:

Aim: Find Peak Element:

Code:

```
class Solution:
    def findPeakElement(self, nums):
        left, right = 0, len(nums) - 1
        while left < right:
            mid = (left + right) // 2
            if nums[mid] > nums[mid + 1]:
                right = mid
            else:
                left = mid + 1
        return left
```

Submission Screenshot:



Ques 5:

Aim: Median of Two Sorted Arrays:

Code:

```
class Solution:
    def findMedianSortedArrays(self, nums1, nums2):
        if len(nums1) > len(nums2):
            nums1, nums2 = nums2, nums1
        x, y = len(nums1), len(nums2)
        left, right = 0, x
        while left <= right:
            partitionX = (left + right) // 2
            partitionY = (x + y + 1) // 2 - partitionX
            maxX = float('-inf') if partitionX == 0 else nums1[partitionX - 1]
            minX = float('inf') if partitionX == x else nums1[partitionX]
            maxY = float('-inf') if partitionY == 0 else nums2[partitionY - 1]
            minY = float('inf') if partitionY == y else nums2[partitionY]
            if maxX <= minY and maxY <= minX:
                if (x + y) % 2 == 0:
                    return (max(maxX, maxY) + min(minX, minY)) / 2.0
                return float(max(maxX, maxY))
            elif maxX > minY:
                right = partitionX - 1
            else:
                left = partitionX + 1
```

Submission Screenshot: