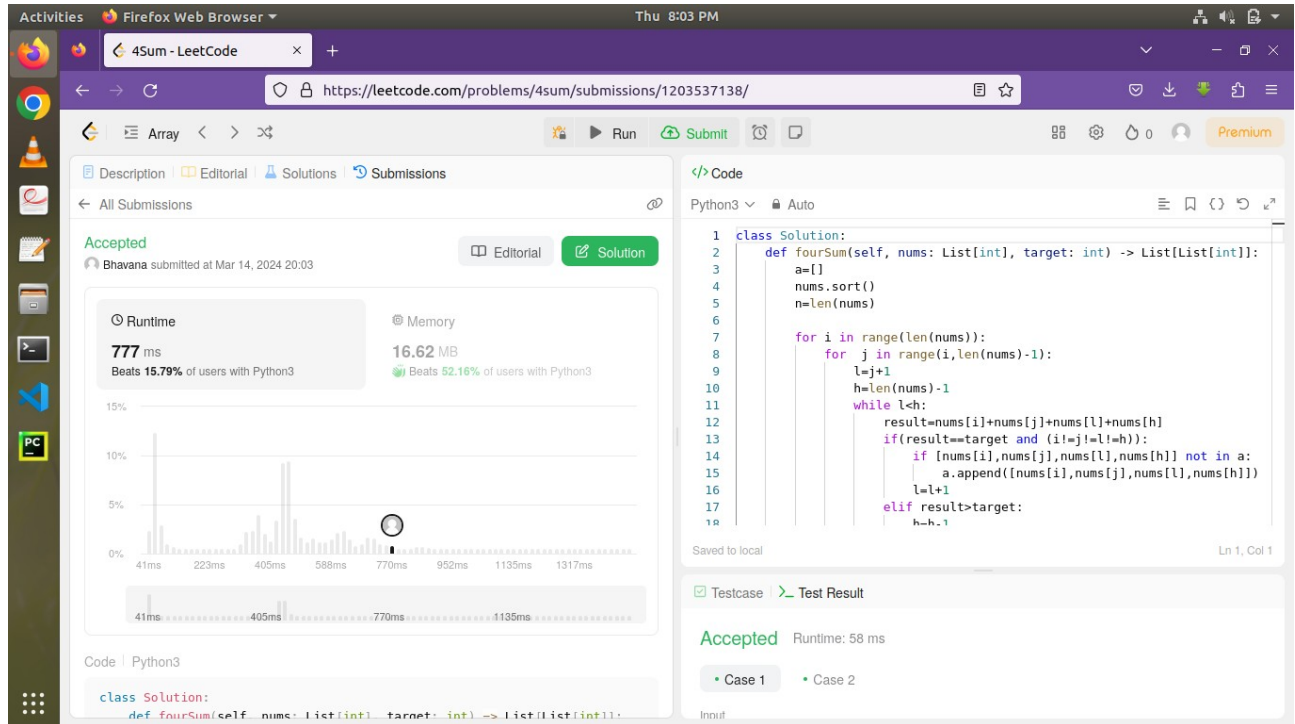


SURE TRUST ASSAIGNMENT

By: Bhavana Annavarm

Q1) 18. 4Sum

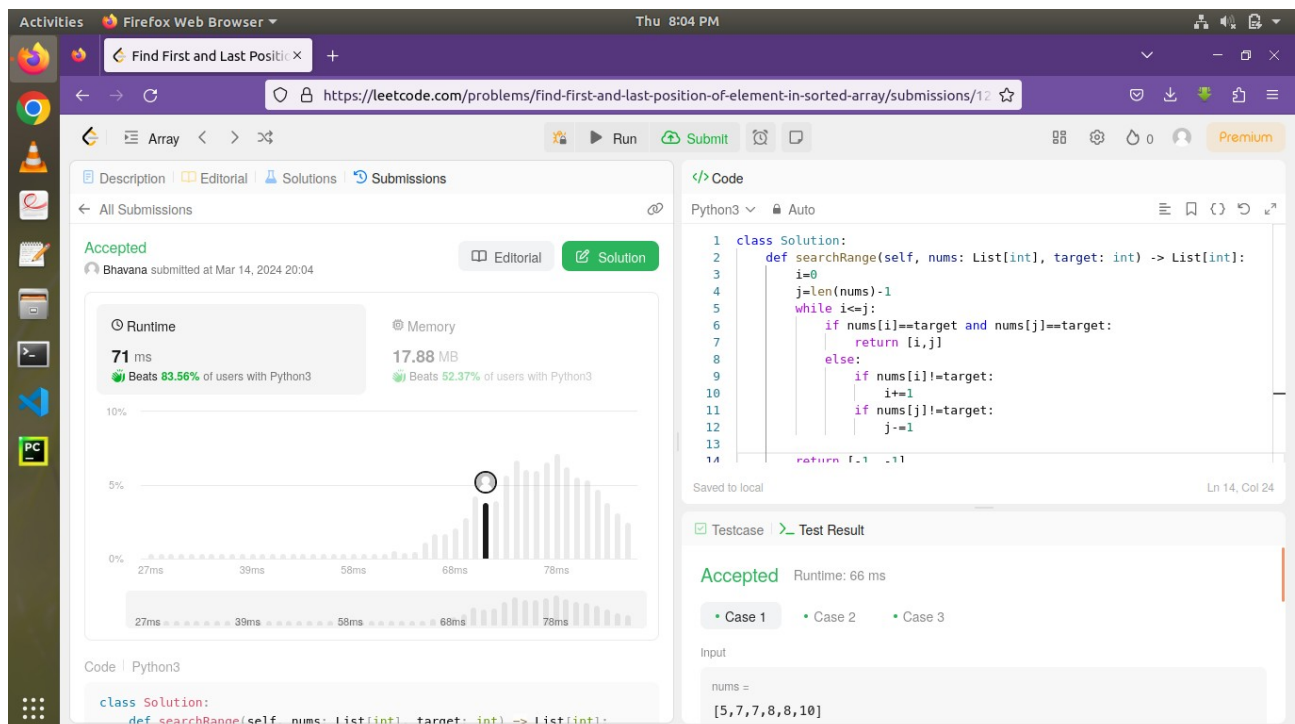


The screenshot shows the LeetCode submission page for the 4Sum problem. The submission is accepted, with a runtime of 777 ms and memory usage of 16.62 MB. The code is written in Python3 and uses a four-pointer approach to find all quadruplets that sum to the target.

```
class Solution:
    def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
        a = []
        nums.sort()
        n = len(nums)

        for i in range(len(nums)):
            for j in range(i, len(nums)-1):
                l = j+1
                h = len(nums)-1
                while l < h:
                    result = nums[i] + nums[j] + nums[l] + nums[h]
                    if result == target and (i, j, l, h) not in a:
                        a.append([nums[i], nums[j], nums[l], nums[h]])
                        l = l+1
                    elif result > target:
                        h = h-1
```

Q2) 34. Find First and Last Position of Element in Sorted Array



The screenshot shows the LeetCode submission page for the Find First and Last Position of Element in Sorted Array problem. The submission is accepted, with a runtime of 71 ms and memory usage of 17.88 MB. The code is written in Python3 and uses a binary search approach to find the first and last positions of the target element.

```
class Solution:
    def searchRange(self, nums: List[int], target: int) -> List[int]:
        i = 0
        j = len(nums)-1
        while i <= j:
            if nums[i] == target and nums[j] == target:
                return [i, j]
            else:
                if nums[i] != target:
                    i += 1
                if nums[j] != target:
                    j -= 1
        return [-1, -1]
```

Q3) 75. Sort Colors

The screenshot shows the LeetCode interface for the 'Sort Colors' problem. The submission is accepted, with a runtime of 31 ms and memory usage of 16.51 MB. The code is a Python3 solution using a nested loop to sort the array in-place.

Runtime: 31 ms (Beats 91.20% of users with Python3)

Memory: 16.51 MB (Beats 48.90% of users with Python3)

Code:

```
class Solution:
    def sortColors(self, nums: List[int]) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        for i in range(len(nums)-1):
            for j in range(i+1, len(nums)):
                if nums[i] > nums[j]:
                    nums[i], nums[j] = nums[j], nums[i]
```

Test Result: Accepted (Runtime: 64 ms)

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

Q4) 81. Search in Rotated Sorted Array II

The screenshot shows the LeetCode interface for the 'Search in Rotated Sorted Array II' problem. The submission is accepted, with a runtime of 51 ms and memory usage of 17.34 MB. The code is a Python3 solution using a linear search to find the target in the rotated array.

Runtime: 51 ms (Beats 67.88% of users with Python3)

Memory: 17.34 MB (Beats 14.23% of users with Python3)

Code:

```
class Solution:
    def search(self, nums: List[int], target: int) -> bool:
        if target in nums:
            return True
        return False
```

Test Result: Accepted (Runtime: 64 ms)

Input: nums = [2, 5, 6, 0, 0, 1, 2]