# MCS – 253P ADVANCED PROGRAMMING AND PROBLEM SOLVING

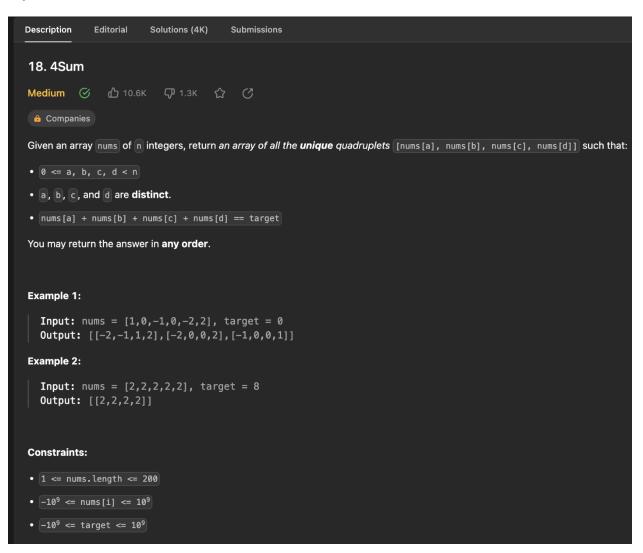
# LAB 2 Write Up (4SUM)

Aswin Sampath saswin@uci.edu

53844684

Date: 15/06/2023

### **Question:**



## Writeup:

#### **Understanding the Problem:**

The problem is to find all unique quadruplets in an array that sums up to a given target. Each quadruplet should consist of distinct elements from the input array.

#### **Identifying Edge Cases:**

Empty Array: We should consider how our code handles the case when the input array nums is empty.

#### **Effective Test Cases:**

- 1. Test Case 1: Input array nums = [1, 0, -1, 0, -2, 2], target = 0. The expected output should be [[ -2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]].
- 2. Test Case 2: Input array nums = [2, 2, 2, 2, 2], target = 8. The expected output should be [[2, 2, 2, 2]]. This test case checks how the code handles a situation where all elements in the array are the same, and it should produce a quadruplet with the same number repeated four times.

#### **Algorithmic Solution:**

- 1. Sort the input array nums in ascending order.
- 2. Iterate Through Pairs:
- 3. Use two nested loops to iterate through pairs of elements: nums[i] and nums[j].
- 4. Calculate twoSumTarget
- 5. For each pair of elements, calculate twoSumTarget as the target minus the sum of the current pair: twoSumTarget = target nums[i] nums[j].
- 6. Find Pairs That Sum to twoSumTarget:
- 7. Use a two-pointer approach with left and right pointers in the subarray nums[j+1:] to find pairs whose sum is equal to twoSumTarget.
- 8. Adjust the left and right pointers accordingly as we search for pairs.
- 9. Construct Quadruplets:
- 10. When a pair (nums[left], nums[right]) is found such that nums[left] + nums[right] equals twoSumTarget, create a quadruplet quad consisting of nums[i], nums[j], nums[left], and nums[right].
- 11. Store Unique Quadruplets:
- 12. Use a vector ans to store the unique quadruplets.
- 13. Ensure that quad is not already in ans before adding it, preventing duplicates from being included in the result.
- 14. Continue Looping:
- 15. Continue the process, updating i and j as well as checking for duplicates, until all unique quadruplets have been found.
- 16. Return the Result:
- 17. Return the ans vector containing all unique quadruplets that sum up to the target.

#### **Time and Space Complexity Analysis:**

**Time Complexity:** Sorting the array takes O(n \* log(n)) time. The code uses nested loops, resulting in an overall time complexity of  $O(n^3)$ . The findTwoSum function uses a two-pointer approach, which takes O(n) time. Therefore, the time complexity of the fourSum function is  $O(n^3)$  (max out of sorting time and nested loops time).

**Space Complexity:** The space complexity is primarily due to the ans vector, the s set, and the allPairs vector. The ans vector may contain up to  $O(n^2)$  quadruplets. The s set may contain up to  $O(n^2)$  unique quadruplets. The allPairs

vector can have a maximum size of $O(n)$ , as it stores pairs of indices. ans and s data structures and $O(n)$ for the allPairs data structure.	Therefore, the space complexity is O(n^2) for the