https://course.acciojob.com/idle?question=8857040e-21ae-471d-b2ee-8b01e18f681d

- MEDIUM
- Max Score: 40 Points

4Sum

Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that:

0 <= a, b, c, d < n

a, b, c, and d are distinct.

nums[a] + nums[b] + nums[c] + nums[d] == target

You may return the answer in any order.

NOTE: You just have to complete the given function.

Expected time Complexity: O(N^3)

Input Format

First line contains the size of the array n.

second line will contain n space separated integers.

third line contains the target

Output Format

Return all the required unique quadruplets.

Example 1

Input

```
6
1 0 -1 0 -2 2
0
```

Output

```
-2 -1 1 2
-2 0 0 2
-1 0 0 1
```

Explanation

There are three unique required quadruplets.

Example 2

Input

```
5
2 2 2 2 2
8
```

Output

```
2 2 2 2
```

Explanation

There is only one unique required quadruplets.

Constraints

1 <= nums.length <= 200

-10^9 <= nums[i] <= 10^9

-10^9 <= target <= 10^9

Topic Tags

- **2-Pointers**
- **Sorting**
- **Arrays**

My code

```
// n java
import java.util.*;
import java.io.*;
class Solution {
   public List<List<Integer>> fourSum(int[] nums, int target) {
       // Write your code here
           List<List<Integer>> ans= new ArrayList<>();
        // Base condition becouse ginven n 1 to 200
        if (nums == null || nums.length < 4) {
            return ans;
        }
        Arrays.sort(nums);// Sort the array
        int n = nums.length;
        for (int i = 0; i < n - 3; i++) {
            // Check for skipping duplicates
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }

            for (int j = i + 1; j < n - 2; j++) {
                // remove duplicates
                if (j != i + 1 && nums[j] == nums[j - 1]) {
                    continue;
                }
                // Left and right pointers k and l
                int k = j + 1;
                int l = n - 1;
                // Reducing to two sum problem
                while (k < l) {
                    int currentSum = nums[i] + nums[j] + nums[k] + nums[l];
                    if (currentSum < target) {
                        k++;
                    } else if (currentSum > target) {
                        l--;
                    } else {//here sum==k
                        ans.add(Arrays.asList(nums[i], nums[j], nums[k], 
nums[l]));
                        k++;
                        l--;
                        // Check for skipping duplicates
                        while (k < l && nums[k] == nums[k - 1]) {
                            k++;
                        }
                        while (k < l && nums[l] == nums[l + 1]) {
                            l--;
                        }
                    }
                }
            }
        }
        return ans;
    }
}
```

```
public class Main
{

    public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
        int n;
        n = sc.nextInt();
        int[] nums = new int[n];
        for (int i = 0; i < n; i++)
            nums[i] = sc.nextInt();
        int k = sc.nextInt();
        Solution Obj = new Solution();
        List<List<Integer>> res = Obj.fourSum(nums, k);


        for(int i= 0; i<res.size(); i++){
           Collections.sort(res.get(i));
        }


        Collections.sort(res, new Comparator<List<Integer>>() {
               public int compare(List<Integer> frst, List<Integer> scnd)
{
                    int i=0;
                    while(frst.get(i)==scnd.get(i)) i++;
                    return frst.get(i)-scnd.get(i);
                    }
                });
        for(int i=0; i<res.size(); i++){
            for(int j=0; j<4; j++){
                System.out.print(res.get(i).get(j) + " ");
            }
            System.out.println("");
        }
        sc.close();
        }
}
```