Day-13

Python DSA

4Sum Problem

https://leetcode.com/problems/4sum/description/

Logic is same as 3Sum Problem

Bruteforce

```
from typing import List
def fourSum( nums: List[int], target: int) -> List[List[int]]:
    n = len(nums)
    if n < 4:
        return []

    my_set = set()  # Using a set to store unique quadruplets

    # Try all possible quadruplets using four nested loops
    for i in range(0, n):
        for j in range(i + 1, n):
            for k in range(j + 1, n):
                for l in range(k + 1, n):
                    total = nums[i] + nums[j] + nums[k] + nums[l]

                    if total == target:  # If sum matches the target
                        temp = [nums[i], nums[j], nums[k], nums[l]]
```

```
                temp.sort()  # Sort to avoid duplicate permutations
                my_set.add(tuple(temp))  # Add as a tuple to the set

    # Convert set to list of lists
    result = []
    for ans in my_set:
        result.append(list(ans))

    return result

nums = [1, 0, -1, 0, -2, 2]
target = 0
fourSum(nums, target)
```

Output

```
[[-2, -1, 1, 2], [-1, 0, 0, 1], [-2, 0, 0, 2]]
```

TC- O(N^4)

SC- O(N^2)

Better

```
def fourSum(nums, target):
    n= len(nums)
    if n<4:
        return []
    my_set=set()
```

```
    for i in range(0,n):
        for j in range(i+1,n):
            hash_set=set()
            for k in range(j+1, n):
                fourth = target - (nums[i] + nums[j] + nums[k])
                if fourth in hash_set:
                    temp = [nums[i] , nums[j] , nums[k] , fourth]
                    temp.sort()
                    my_set.add(tuple(temp))
                hash_set.add(nums[k])
    result= [list(ans) for ans in my_set]
    return result
nums = [1, 0, -1, 0, -2, 2]
target = 0
fourSum(nums, target)

nums = [1, 0, -1, 0, -2, 2]
target = 0
fourSum(nums, target)
```

Output

[[-2, -1, 1, 2], [-1, 0, 0, 1], [-2, 0, 0, 2]]

TC- O(N^3) =O(N^2)=O(N^3)

SC – O(N^2)

Optimal

```python
class Solution:

    def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
        n = len(nums)
        ans = []
        nums.sort()  # Step 1: Sort the array

        # Step 2: First two loops to pick the first two numbers
        for i in range(0, n):
            if i > 0 and nums[i] == nums[i - 1]:  # Skip duplicate elements
                continue

            for j in range(i + 1, n):
                if j > i + 1 and nums[j] == nums[j - 1]:  # Skip duplicate elements
                    continue

                # Step 3: Two-pointer approach for remaining two numbers
                k = j + 1  # Left pointer
                l = n - 1  # Right pointer

                while k < l:
                    total = nums[i] + nums[j] + nums[k] + nums[l]

                    if total == target:
                        ans.append([nums[i], nums[j], nums[k], nums[l]])
                        k += 1
                        l -= 1

                        # Skip duplicate elements for k and l
                        while k < l and nums[k] == nums[k - 1]:
                            k += 1
                        while l > k and nums[l] == nums[l + 1]:
                            l -= 1

                    elif total < target:
                        k += 1  # Move left pointer to increase sum

                    else:
```

```
                l -= 1  # Move right pointer to decrease sum

        return ans
```

TC – O(NlogN) + O(N^2 * N)= O(N^3)

SC- O(N^2)