18. 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**.

Example 1:

Input: nums = [1,0,-1,0,-2,2], target = 0
Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

Example 2:

Input: nums = [2,2,2,2,2], target = 8
Output: [[2,2,2,2]]
My Approach:

```
def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
        nums.sort()
        n = len(nums)
        res = []
        for i in range(n-3):
            if i!=0 and nums[i]==nums[i-1]:
                continue
            for j in range(i+1,n-2):
                if j!=i+1 and nums[j]==nums[j-1]:
                    continue
                st = j+1
                en = n-1
                while st<en:
                    sum = nums[i]+nums[j]+nums[st]+nums[en]
                    if sum>target:
                        en = en-1
                    elif sum<target:
                        st = st+1
                    else:
```

```python
                        res.append([nums[i],nums[j],nums[st],nums[en]])
                        st = st+1
                        en = en-1

                        while st<en and nums[st]==nums[st-1]:
                            st = st+1
                        while st<en and nums[en]==nums[en+1]:
                            en = en-1
        return res
```