15. 3Sum

Given an integer array nums, return all the triplets `[nums[i], nums[j], nums[k]]` such that `i != j`, `i != k`, and `j != k`, and `nums[i] + nums[j] + nums[k] == 0`.

Notice that the solution set must not contain duplicate triplets.

Example 1:

```
Input: nums = [-1,0,1,2,-1,-4]
Output: [[-1,-1,2],[-1,0,1]]
```

Example 2:

```
Input: nums = []
Output: []
```

Example 3:

```
Input: nums = [0]
Output: []
```

Constraints:

- `0 <= nums.length <= 3000`
- `-10<sup>5</sup> <= nums[i] <= 10<sup>5</sup>`
- ```
  class Solution:
      def threeSum(self, nums: List[int]) -> List[List[int]]:
          nums.sort()


          res = []
          n = len(nums)
          for i in range(n):
              if i>0 and nums[i]==nums[i-1]:
                  continue
              else:
                  target = 0-nums[i]
                  temp = self.twoSums(nums,i+1,n-1,target)
                  if len(temp)>0:
                      temp = self.result(nums[i],temp)
              res = res+temp
  ```
  ```

```
        return res

    def result(self,ele,ans):
        for i in range(len(ans)):
            ans[i] = ans[i]+[ele]
        return ans




    def twoSums(self,nums,start,end,target):
        ans = []
        while start<end:
            temp = nums[start]+nums[end]
            if temp == target:
                ans.append([nums[start],nums[end]])
                start+=1
                end-=1

                while start<end and nums[start]==nums[start-1]:
                    start+=1
                while start<end and nums[end]==nums[end+1]:
                    end-=1
            elif temp>target:
                end-=1
            else:
                start+=1

        return ans
```