3 Sum

Solve the 3 Sum problem - a well-known classic involving sorting and the two-pointer technique.

Key Idea: You want to find all unique triplets (i, j, k) such that: nums[i] + nums[j] + nums[k] == 0
To do this efficiently, avoid brute force ($O(n^3)$)

Algorithm (Two-pointer + Sorting):

1. Sort the input array nums.
2. ~~Loop~~ through each element nums[i]:
   - For each i, use two pointers: left = i+1, right = n-1
   - While left < right:
     - compute sum = nums[i] + nums[left] + nums[right]
     - if sum == 0, store the triplet and skip duplicates
     - if sum < 0, move left++
     - if sum > 0, move right --
3. Skip duplicate values of i, left and right to avoid repeated triplets.

Why Sorting Helps:
- It makes duplicate detection easy.
- It allows the two-pointer technique to work (since you're checking sum relations).

Important Edge Cases:
- [0, 0, 0] → Only one triplet [0, 0, 0]
- duplicate numbers → skip over them using a while loop.
- result must not include duplicate triplets.

Time Complexity:
- sorting: O(n log n)
- two pointers: $O(n^2)$ overall

This is the optimal solution for this problem