LEETCODE | 15 | 18 DATE

3 SUM | 4 SUM

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

To have an optimal approach where indices are not needed we can sort the array

Step1 Sort the array → Arrays.sort(nums)

↓

nums = [-4, -1, -1, 0, 1, 2]

Step2 Use 3 pointers

[-4, -1, -1, 0, 1, 2]

i=0 i j=i+1 k=nums.length-1

To optimize

Imp if (i>0 && nums[i]=nums[i-1]) { continue } → skip to next i

Step3 Keep i as constant & move j & k such that j<k

int sum = nums[i] + nums[j] + nums[k]

if (sum < 0) {//increase value → j++}

else if (sum > 0) {//decrease value → k--}

else { // Save the triplet here}

↑ When triplet is added to ~~resultant~~ List of list

j++, k--;

To optimise it; check (→ 1) while (j<k && nums[j] == nums[j-1]) { j++; }

Imp

2) while (j<k && nums[k] == nums[k+1]) { k--; }

Note:
Asum approach is same.

classmate PAGE