Three Sum

15. 3Sum

Solved

Medium | Topics | Companies | Hint

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]]
Explanation:
nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.
nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.
nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.
The distinct triplets are [-1,0,1] and [-1,-1,2].
Notice that the order of the output and the order of the triplets does not matter.
```

Example 2:

```
Input: nums = [0,1,1]
Output: []
Explanation: The only possible triplet does not sum up to 0.
```

Example 3:

```
Input: nums = [0,0,0]
Output: [[0,0,0]]
Explanation: The only possible triplet sums up to 0.
```

problem Statement

* if we add 3 different indices values, that will equal 0
* if that happens, store such index values triplet and return all triplets
* No Duplicates are allowed

Suppose :- {-1, 0, 1, 2, 2, -1, 4}

ans1 = {-1, 0, 1, 2, 2, -1, 4}

ans2 = {-1, 0, 1, 2, 2, -1, 4}

Both results in zero, But both triplets are Same, So don't allow duplicates

For that to happen, Sort the elements and skip duplicates

And if we have a Sorted array, and we are looking for a certain condition on index values,
we can use two pointers technique

nums[i] + nums[j] + nums[k] == 0

nums[i] = {-1, 0, 1, 2, -1, 4}

⇓ sort

{-1, -1, 0, 1, 2, 4}

placing pointers

{-1, -1, 0, 1, 2, 4}

Initially :

i j k

* We start with i, which traverse from 0 → n - 1

* for every i, initialize 2 pointers (j, k)

j at i+1, k at n-1

(from front) (from Back)

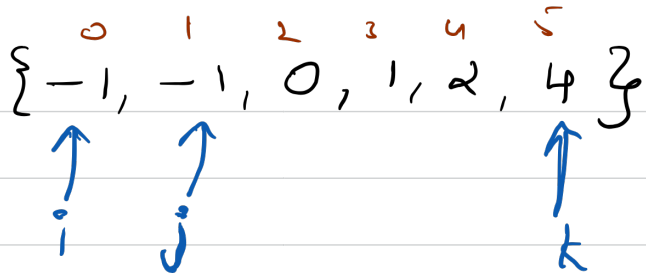
* And apply 2sum ideology b/w (J,K) pointers

for every (j, k) look Up for nums[i] + nums[j] + nums[k] == 0?

* if (sum < 0) → As array is sorted, we have to increase sum Value, increment j

* if (sum > 0) → As array is sorted, we have to decrease sum Value, decrement k

* if (sum == 0) → Triplet found with, add it to ans & move (j,k)



i	j	k	Sum	Operation
0	1	5	$-1-1+4=2$	$2>0$ (k--)
0	1	4	$-1-1+2=0$	add this triplet

// Skip all k = 2's & Skip all j == -1's

0	2	3	$-1+0+1=0$	add this triplet

// Skip all k = 1's & Skip all j == 0's

if (k & j) Crosses → increment & do the same process again

Because, with i=0, you can't form any other triplets with same values of (j, k) → So increment j, decrement k } till we get new values

* Above Steps are for triplets which includes $i=0$

* Do Same Steps for $i \in 1, 2, 3, \ldots n$

* Then our ans will consists of every possible triplet.

Time Complexity

① Sorting the array $\rightarrow O(n \log n)$

② nearly $O(n^2)$ to traverse and check all triplets

$\therefore \approx O(n^2)$