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.
```

* if we add 3 elements index values, that will be equal to 0, then

↓

if this happens, store such index values triplet and return all triplets

* No duplicates allowed

Suppose : (-1, 0, 1, 2, 2, -1, 4)

ans1 = (-1, 0, 1, 2, 2, -1, 4) (-1, 0, 1, 2, 2, -1, 4) = ans2

Both answers results in zero, but both triplets are same, so don't allow triplets

Simply, for that to happen...

① Sort the elements

② 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

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

nums[] : (-1, 0, 1, 2, -1, 4)

↓ sort

(-1, -1, 0, 1, 2, 4)

placing pointers :

(-1, -1, 0, 1, 2, 4)

↑ i ↑ j ↑ k

* We start with i → traverse from 0 to n-2

* for every i → we have (j, k)
  ↳ j starts from i+1
  k starts from n-1

Now, apply 2 Sum ideology, b/w (j, k) pointers

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

if (Sum < 0) → j++ { Sum need to increase, so increment j, as array Sorted value will increase

if (Sum > 0) → k-- { Sum need to decrease }

if (Sum == 0) → Add triplet & move j, k

* When (j, k) crosses, move (i)

arr : (-1, -1, 0, 1, 2, 4)
  0   1   2 3 4 5

↑ i ↑ j ↑ k

	i	j	k	Sum	Operation
arr : (-1, -1, 0, 1, 2, 4) ↑i ↑j ↑k	0	1	5	2 > 0	k--
arr : (-1, -1, 0, 1, 2, 4) ↑i ↑j ↑k	0	1	4	0 == 0	Add triplet
// Skip all k = 2's & j = -1's to ignore duplicates					
arr : (-1, -1, 0, 1, 2, 4) ↑i ↑j ↑k	0	2	3	0 == 0	Add triplet

Because, with i=0, you can't form any other triplets with same values of (j, k)

if (j & k Crosses)
↳
increment i → & repeat same

* Above steps are for triplets which includes (i=0)

* Do same steps for i=1, 2, ... n-2

* Then our ans will consists of every possible triplet

T.C: O(nlogn) → Sorting

O(n²) to traverse (i, j, k) ≈ O(n²)