Description Solution Discuss (999+) Submissions

15. 3Sum

Medium 21935 2011 Add to List Share

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.
```

Constraints:

- `3 <= nums.length <= 3000`
- `-10^5 <= nums[i] <= 10^5`

Java Autocomplete

```java
class Solution {

    public  List<List<Integer>> threeSum(int[] nums) {

        Set<List<Integer>> res  = new HashSet<>();

        if(nums.length==0)
            return new ArrayList<>(res);

        Arrays.sort(nums);

        for(int i = 0; i < nums.length-2; i++){

            int j = i+1;
            int  k = nums.length-1;

            while(j < k){
                int sum = nums[i] + nums[j] + nums[k];

                if(sum==0)
                    res.add(Arrays.asList(nums[i], nums[j++], nums[k--]));

                else if ( sum >0)
                    k--;
                else if (sum<0)
                    j++;
            }

        }

        return new ArrayList<>(res);
    }
}
```

Testcase Run Code Result Debugger

Accepted Runtime: 0 ms

Your input `[-1,0,1,2,-1,-4]`

Output `[[-1,-1,2],[-1,0,1]]` Diff

Expected `[[-1,-1,2],[-1,0,1]]`