# 15. 3Sum

Description | Hints | Submissions | Discuss | Solution

Pick One

Given an array `nums` of *n* integers, are there elements *a*, *b*, *c* in `nums` such that *a* + *b* + *c* = 0? Find all unique triplets in the array which gives the sum of zero.

**Note:**

The solution set must not contain duplicate triplets.

**Example:**

```
Given array nums = [-1, 0, 1, 2, -1, -4],

A solution set is:
[
  [-1, 0, 1],
  [-1, -1, 2]
]
```

```java
public class L15 {
      public List<List<Integer>> threeSum(int[] nums) {
        //首先要返回什么就创建什么
          List<List<Integer>> ret = new ArrayList<>();
          if(nums == null || nums.length < 3)  //边界条件判断
              return ret;
          int len = nums.length;
          Arrays.sort(nums);   //对数组按照从小到大的顺序进行排列
         //对于num[i],寻找另外两个数时，只要从i+1开始找就可以了，这种写法，可以避免结果中有重复，因为数组是排好序的，所以当一个数被放到结果集中的时候，后i
          for(int i = 0; i < len; i++) {
              if(nums[i] > 0)  //这个的意思是如果第一个元素都大于0.那么排好序后的这三个数都大于0，所以不可能出现加起来为0的情况
                  break;
              if(i > 0 && nums[i] == nums[i - 1]) //这儿是避免重复
                  continue;
              int begin = i + 1;  //往后找，避免重复
              int end = len - 1;
              while (begin < end) {
                int sum = nums[i] + nums[begin] + nums[end];
                if (sum == 0) {
                    List<Integer> list = new ArrayList<>();
                    list.add(nums[i]);
                    list.add(nums[begin]);
                    list.add(nums[end]);
                    ret.add(list);
                    begin ++;
                    end --;
                    while (begin < end && nums[begin] == nums[begin - 1]) {//这儿也是避免重复，因为begin加1了，end减1了，
                        begin ++;
                    }
                    while (begin < end && nums[end] == nums[end + 1]) {
                        end --;
                    }
                }else if (sum > 0) {
                    end --;
                }else
                    begin ++;
            }
          }
        return ret;
      }
}
```