三数之和

> 思路其实和答案一样，逻辑也是和答案相似的，但是就是不知道为什么有些用例跑不过

> 思路：
>
> 将三数之和为0转化为两数之和等于第三个数
>
> 第一层循环，遍历每一个元素，作为第三个数
>
> 第二层循环，利用双指针，找到两数和等于第三个元素。
>
> 但是我的代码在处理重复上没什么逻辑

1、my-code（能跑了，但是改了无数遍，有点像面向OJ编程了）

```
vector<vector<int> > threeSum(vector<int>&nums){
    if (nums.size() < 3) return {};
    sort(nums.begin(), nums.end());
    vector<vector<int> > res;
    for (int i = 0; i < nums.size(); ++i) {
        if (i > 0 && nums[i] == nums[i-1]) continue;
        //if ((nums[0] == 0 && nums[i] > 0) || (nums[0] < 0 && nums[i] >= 0)) break;
        int c = -1 * nums[i];
        int j = i + 1, k = nums.size() - 1;
        while (j < k){
//             if (j == i) j++;
//             if (k == i) k--;
            if (nums[j] + nums[k] == c){
                res.push_back({-1*c, nums[j], nums[k]});
//                 if (nums[j] == nums[j + 1]) break;
                j++;k--;
            } else if (nums[j] + nums[k] > c){
                k--;
            } else {
                j++;
            }
            while (k < nums.size()-1 && k >= 0 && nums[k] == nums[k+1]){
                k--;
            }
            while (j > i + 1 && j < nums.size() && nums[j] == nums[j-1]){
                j++;
            }
        }
    }
    return res;
}
```

2、official-code

```cpp
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        int n = nums.size();
        sort(nums.begin(), nums.end());
        vector<vector<int>> ans;
        // 枚举 a
        for (int first = 0; first < n; ++first) {
            // 需要和上一次枚举的数不相同
            if (first > 0 && nums[first] == nums[first - 1]) {
                continue;
            }
            // c 对应的指针初始指向数组的最右端
            int third = n - 1;
            int target = -nums[first];
            // 枚举 b
            for (int second = first + 1; second < n; ++second) {
                // 需要和上一次枚举的数不相同
                if (second > first + 1 && nums[second] == nums[second - 1])
{
                    continue;
                }
                // 需要保证 b 的指针在 c 的指针的左侧
                while (second < third && nums[second] + nums[third] >
target) {
                    --third;
                }
                // 如果指针重合，随着 b 后续的增加
                // 就不会有满足 a+b+c=0 并且 b<c 的 c 了，可以退出循环
                if (second == third) {
                    break;
                }
                if (nums[second] + nums[third] == target) {
                    ans.push_back({nums[first], nums[second], nums[third]});
                }
            }
        }
        return ans;
    }
};
```