15. 3Sum

Medium 9154 959 Add to List Share

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.

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]]
```

Example 2:

```
Input: nums = []
Output: []
```

Example 3:

```
Input: nums = [0]
Output: []
```

Constraints:

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

```
vector<vector<int>> twoSumTarget(vector<int>& nums, int target) {
    sort(nums.begin(), nums.end());
    int lo = 0, hi = nums.size() - 1;
    vector<vector<int>> res;
    while (lo < hi) {
        int sum = nums[lo] + nums[hi];
        int left = nums[lo], right = nums[hi];
        if (sum < target) {
            while (lo < hi && nums[lo] == left) lo++;
        } else if (sum > target) {
            while (lo < hi && nums[hi] == right) hi--;
        } else {
            res.push_back({left, right});
            while (lo < hi && nums[lo] == left) lo++;
            while (lo < hi && nums[hi] == right) hi--;
        }
    }
    return res;
}
```

Time complexity: $O(N \log N)$

3 sum:

1. For the first number, use exhaustion
2. Use twoSumTarget(nums[i+1, -1], target - nums[i])

Time complexity: $O(N^2)$

```cpp
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        vector<vector<int>> res;
        for (int i = 0; i < nums.size(); i++) {
            vector<vector<int>> tuples = twoSumTarget(nums, i + 1, -nums[i]);
            for (auto tuple : tuples) {
                tuple.push_back(nums[i]);
                res.push_back(tuple);
            }
            while (i < nums.size() - 1 && nums[i] == nums[i + 1]) {
                i++;
            }
        }
        return res;
    }

private:
    vector<vector<int>> twoSumTarget(vector<int>& nums, int start, int target)
{
        int lo = start, hi = nums.size() - 1;
        vector<vector<int>> res;
        while(lo < hi) {
            int left = nums[lo], right = nums[hi];
            int sum = left + right;
            if (target > sum) {
                while (lo < hi && nums[lo] == left) {
                    lo++;
                }
            } else if (target < sum) {
                while (lo < hi && nums[hi] == right) {
                    hi--;
                }
            } else {
                res.push_back({left, right});
                while (lo < hi && nums[lo] == left) {
                    lo++;
                }
                while (lo < hi && nums[hi] == right) {
                    hi--;
                }
            }
        }
        return res;
    }
};
```