

# 018. 4Sum

## 018 4Sum

- Hash Table
- Two Pointers

### Description

Given an array  $S$  of  $n$  integers, are there elements  $a, b, c$ , and  $d$  in  $S$  such that  $a + b + c + d = \text{target}$ ? Find all unique quadruplets in the array which gives the sum of target.

**Note:** The solution set must not contain duplicate quadruplets.

For example, given array  $S = [1, 0, -1, 0, -2, 2]$ , and  $\text{target} = 0$ .

A solution set is:

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

### 1. Thought line

### 2 Two Pointers

#### 2.1 Without optimization

```
1 class Solution {
2 public:
3     vector<vector<int>> fourSum(vector<int>& nums, int target) {
4         vector<vector<int>> result(0);
5         if (nums.empty() || nums.size() < 4) return result;
6         sort(nums.begin(), nums.end());
7         int a = 0, b = 0, c = 0, d = 0;
8         for (int i = 0; i <= nums.size() - 4; ++i) {
9             for (int j = i + 1; j <= nums.size() - 3; ++j) {
10                for (int k = j + 1; k <= nums.size() - 2; ++k) {
11                    for (int p = k + 1; p <= nums.size() - 1; ++p) {
12                        if (target == (nums[i] + nums[j] + nums[k] + nums[p])) {
13                            {
14                                result.push_back({nums[i], nums[j], nums[k], nums[p]});
15                            }
16                            while (p + 1 <= nums.size() - 1 && nums[p + 1] == nums[p])
17                                ++p;
18                        }
19                        while (k + 1 <= nums.size() - 2 && nums[k + 1] == nums[k])
20                            ++k;
21                    }
22                    while (j + 1 <= nums.size() - 3 && nums[j + 1] == nums[j])
23                        ++j;
24                }
25                while (i + 1 <= nums.size() - 4 && nums[i + 1] == nums[i])
26                    ++i;
27            }
28            return result;
29        }
30    };
```

#### 2.2 Two Pointers with optimization

```

1 # 12 ms
2 class Solution {
3 public:
4     vector<vector<int>> fourSum(vector<int>& nums, int target) {
5         vector<vector<int>> total;
6         int n = nums.size();
7         if(n<4) return total;
8         sort(nums.begin(),nums.end());
9
10        for(int i=0;i<n-3;i++)
11        {
12            // move forward if it's duplicate number
13            //if(i>0&&nums[i]==nums[i-1]) continue;
14
15            // jump out
16            if(nums[i]+nums[i+1]+nums[i+2]+nums[i+3]>target) break;
17            // need bigger(new) nums[i]
18            if(nums[i]+nums[n-3]+nums[n-2]+nums[n-1]<target) continue;
19
20            for(int j=i+1;j<n-2;j++)
21            {
22                if(j>i+1 && nums[j]==nums[j-1]) continue;
23
24                if(nums[i]+nums[j]+nums[j+1]+nums[j+2]>target) break;
25                if(nums[i]+nums[j]+nums[n-2]+nums[n-1]<target) continue;
26
27                int left=j+1,right=n-1;
28                while(left<right){
29                    int sum=nums[left]+nums[right]+nums[i]+nums[j];
30                    if(sum<target) left++;
31                    else if(sum>target) right--;
32                    else{
33                        total.push_back(
34                            vector<int>>{nums[i],nums[j],nums[left],nums[right]});
35                        do{left++;}while(nums[left]==nums[left-1]&&left<right);
36                        do{right--;}while(nums[right]==nums[right+1]&&left<right);
37                    }
38                }
39            }
40            while(i+1<n-3 && nums[i+1]==nums[i])
41                ++i;
42        }
43        return total;
44    }
45 };
46
47 /*
48 1. make limited condition
49 use nums[i] as the core
50 (1) if sum of nums{[i,...,i+n]} > target
51 finish
52 (2) if sum of nums{[i-n,...,i]}< target
53 need bigger ones
54
55 2. left and right pointers to perform approximationss
56
57 */

```

### 3.Hash Table