

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

```
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        vector<vector<int>> result(0);
        if (nums.empty() || nums.size() < 4) return result;
        sort(nums.begin(), nums.end());
        int a = 0, b = 0, c = 0, d = 0;
        for (int i = 0; i <= nums.size() - 4; ++i) {
            for (int j = i + 1; j <= nums.size() - 3; ++j) {
                for (int k = j + 1; k <= nums.size() - 2; ++k) {
                    for (int p = k + 1; p <= nums.size() - 1; ++p) {
                        if (target == (nums[i] + nums[j] + nums[k] + nums[p])) {
                            result.push_back({nums[i], nums[j], nums[k], nums[p]});
                        }
                        while (p + 1 <= nums.size() - 1 && nums[p + 1] == nums[p])
                            ++p;
                    }
                    while (k + 1 <= nums.size() - 2 && nums[k + 1] == nums[k])
                        ++k;
                }
                while (j + 1 <= nums.size() - 3 && nums[j + 1] == nums[j])
                    ++j;
            }
            while (i + 1 <= nums.size() - 4 && nums[i + 1] == nums[i])
                ++i;
        }
    }
}
```

```

        return result;
    }
};

```

2.2 Two Pointers with optimization

```

# 12 ms
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        vector<vector<int>> total;
        int n = nums.size();
        if(n<4) return total;
        sort(nums.begin(),nums.end());

        for(int i=0;i<n-3;i++)
        {
            // move forward if it's duplicate number
            //if(i>0&&nums[i]==nums[i-1]) continue;

            // jump out
            if(nums[i]+nums[i+1]+nums[i+2]+nums[i+3]>target) break;
            // need bigger(new) nums[i]
            if(nums[i]+nums[n-3]+nums[n-2]+nums[n-1]<target) continue;

            for(int j=i+1;j<n-2;j++)
            {
                if(j>i+1 && nums[j]==nums[j-1]) continue;

                if(nums[i]+nums[j]+nums[j+1]+nums[j+2]>target) break;
                if(nums[i]+nums[j]+nums[n-2]+nums[n-1]<target) continue;

                int left=j+1,right=n-1;
                while(left<right){
                    int sum=nums[left]+nums[right]+nums[i]+nums[j];
                    if(sum<target) left++;
                    else if(sum>target) right--;
                    else{
                        total.push_back(
                            vector<int>{nums[i],nums[j],nums[left],nums[right]});
                        do{left++;}while(nums[left]==nums[left-1]&&left<right);
                        do{right--;}while(nums[right]==nums[right+1]&&left<right);
                    }
                }
            }
            while(i+1<n-3 && nums[i+1]==nums[i])
                ++i;
        }
        return total;
    }
};

/*
1. make limited condition
use nums[i] as the core
(1) if sum of nums{[i,...,i+n]} > target
finish
(2) if sum of nums{[i-n,...,i]}< target
need bigger ones

2. left and right pointers to perform approximationss
*/

```

3.Hash Table

