4Sum

Question Solution My Submissions (/problems/4sum/submissions/)

Total Accepted: **46927** Total Submissions: **215115** Difficulty: **Medium**

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

Note:

- Elements in a quadruplet (*a*,*b*,*c*,*d*) must be in non-descending order. (ie, *a* ≤ *b* ≤ *c* ≤ *d*)
- The solution set must not contain duplicate quadruplets.

```
    For example, given array S = {1 0 -1 0 -2 2}, and target = 0.

    A solution set is:
    (-1,  0, 0, 1)
    (-2, -1, 1, 2)
    (-2,  0, 0, 2)
```

Show Tags

Show Similar Problems

Have you met this question in a real interview? Yes No

Discuss (/discuss/questions/oj/4sum)

C++ ▼

```
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        vector<vector<int>> res;
        int length=nums.size();
        if(length<4)return res;
        sort(nums.begin(),nums.end());
        for(int i=0;i<length-3;i++){
            if(i>0 && nums[i]==nums[i-1])continue;
            for(int j=i+1;j<length-2;j++){
                if(j>i+1 && nums[j]==nums[j-1])continue;
                int begin=j+1,end=length-1;
                while(begin<end){
                    int sum=nums[i]+nums[j]+nums[begin]+nums[end];
                    if(sum==target){
                        vector<int> temp;
                        temp.push_back(nums[i]);
                        temp.push_back(nums[j]);
```

✉ Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

```
                        temp.push_back(nums[begin]);
                        temp.push_back(nums[end]);
                        res.push_back(temp);
                        begin++;end--;
                        while(begin<end && nums[begin]==nums[begin-1])begin+
                        while(begin<end && nums[end]==nums[end+1])end++;
                    }else if(sum>target){
                        end--;
                        while(begin<end && nums[end]==nums[end+1])end--;
                    }else{
                        begin++;
                        while(begin<end && nums[begin]==nums[begin-1])begin+
                    }
                }
            }
        }
        return res;
    }
};
```

Custom Testcase □

Run Code Submit Solution

Frequently Asked Questions (/faq/) | Terms of Service (/tos/)

Privacy

Copyright © 2015 LeetCode