35. 4 Sum | Brute - Better - Optimal with Codes

18. 4Sum

Medium 9082 1075 Add to List Share

Given an array `nums` of `n` integers, return *an array of all the* ***unique*** *quadruplets* `[nums[a], nums[b], nums[c], nums[d]]` such that:

- `0 <= a, b, c, d < n`
- `a, b, c,` and `d` are **distinct**.
- `nums[a] + nums[b] + nums[c] + nums[d] == target`

You may return the answer in **any order**.

Example 1:

```
Input: nums = [1,0,-1,0,-2,2], target = 0
Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]
```

Example 2:

```
Input: nums = [2,2,2,2,2], target = 8
Output: [[2,2,2,2]]
```

nums[i] + nums[j] + nums[k] = target

[i != j != k != l]

arr[] = [1 0 -1 0 -2 2] target = 0

o/p => [-2 -1 1 2] [-2 0 0 2] [-1 0 0 1]

1) Brute force:

- four loop i, j, k l

```cpp
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        int n = nums.size();
        set<vector<int>> st;
        for(int i = 0;i<n;i++){
            for(int j = i+1;j<n;j++){
                for(int k = j+1;k<n;k++){
                    for(int l = k+1;k<n;k++){
                        long long sum = nums[i] + nums[j];
                        sum += nums[k];
                        sum += nums[l];
                        if(sum == target){
                            vector<int> temp = {nums[i], nums[j], nums[k], nums[l]};
                            sort(temp.begin(), temp.end());
                            st.insert(temp);
                        }
                    }
                }
            }
        }

        vector<vector<int>> ans(st.begin(),st.end());
        return ans;
    }
};
```

x

T.C $\Rightarrow O(n^4)$ S.C $\Rightarrow$ 2 x O(no of quads)

2) Better Solution

$n^4 \simeq n^3$

arr[] = [1 0 -1 0 -2 2]

nums[l] = target - (num[i] + num[j] + num[k])

three loops will generate triplet.

Suppose target = 0 and, i = 1, j = 0, k = 0

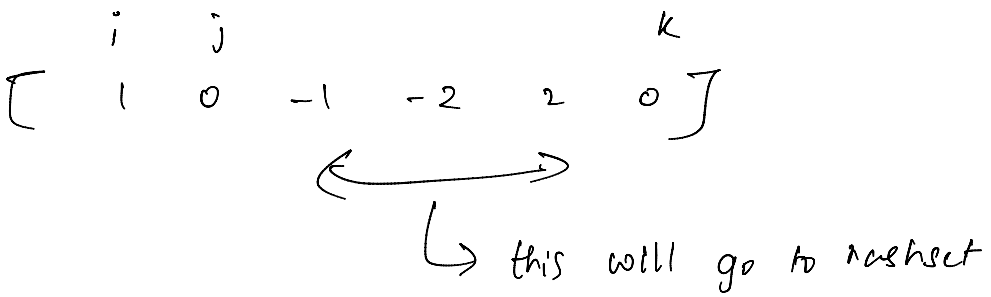
Then

nums[i] = 0 - (1 + 0 + 0)

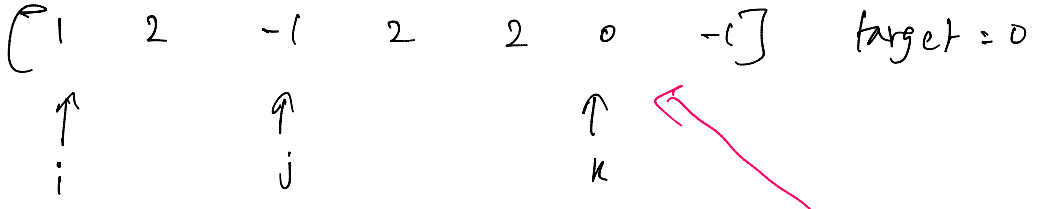
= 0 - 1

nums[i] = -1 // You are looking '-1' in the array if it present then it's quadr.

Similarly to 3-sum.



Not the entire array because suppose



nums[i] = 0 - (1 - 1 + 0)

= 0

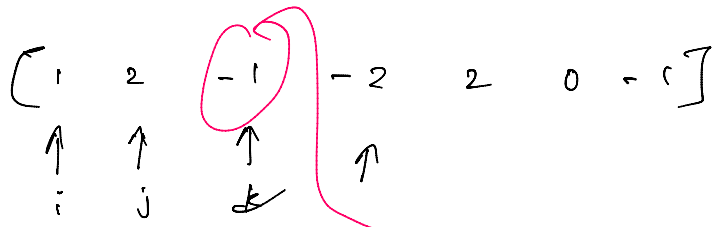
but you will end up picking up same element

and as per question you need to take distinct

value so to avoid that we keep

i = at first place

j = i+1 k = j+1



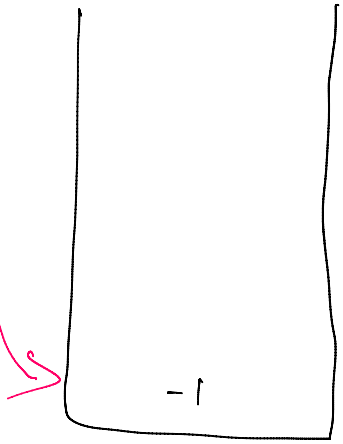
now we start moving the 'k',

Once you move 'k', take it to

the hash set and check

1 + 2 + (-2) is 4th element in the hash set

or not.



So everything b/w j and k will be in hashset

```cpp
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        int n = nums.size();
        set<vector<int>> st;
        for(int i = 0;i<n;i++){
            for(int j = i+1;j<n;j++){
                set<long long> hashset;
                for(int k = j+1;k<n;k++){
                    long long sum = nums[i] + nums[j];
                    sum += nums[k];
                    long long fourth = target - (sum);
                    if(hashset.find(fourth) != hashset.end( )){
                        vector<int> temp = {nums[i], nums[j], nums[k], (int)fourth};
                        sort(temp.begin(), temp.end());
                        st.insert(temp);
                    }
                    hashset.insert(nums[k]);
                }
            }
        }

        vector<vector<int>> ans(st.begin(),st.end());
        return ans;
    }
};
```

T.C ⇒ $O(n^3 \times \log(m))$ S.C ⇒ $O(n) + O(\text{quads}) \times 2$

3rd Optimal Solution :

target = 8

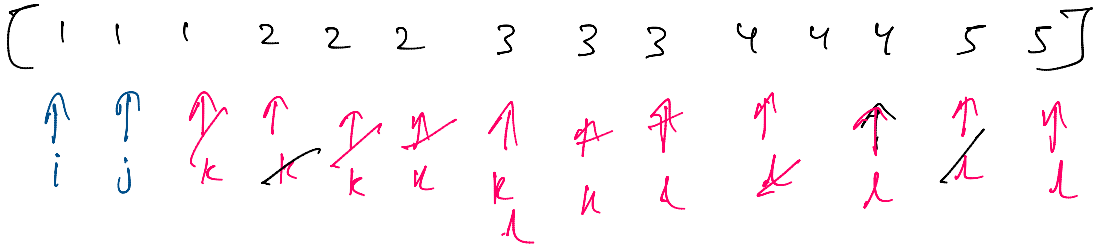
arr[] = [1 1 1 2 2 2 3 3 3 4 4 4 5 5]

in order to have looping correct we sort the array and to avoid duplicate

i j k l,

so try to fix two things as two pointer and two

so try to fix two things as two pointer and two will go constants. So fix i and j.



1+1+1+5 = 8 one quad

now move k and l, but in l it's '5' and for that you need '1' in k and it become duplicate as take same element in k, so move 'l' to avoid taking '5'.

1 + 1 + 4 + 2 = 8 // 2nd quad

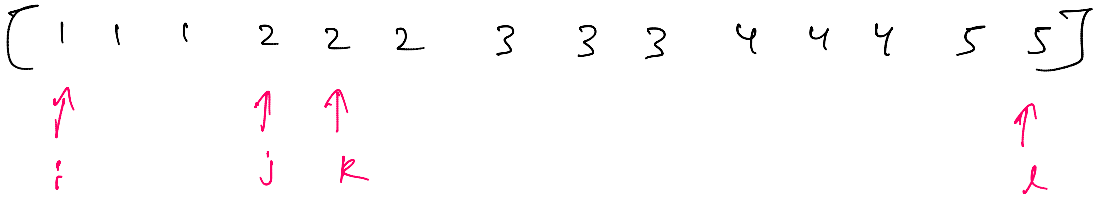
again while moving we get

1+1+3+3 = 8

once L cross k, you stop.

now again move j until possible possible point

which is not similar.



we will do the same thing.

Make sure to avoid duplicates by comparing it with the previous.

```cpp
#include <bits/stdc++.h>
vector<vector<int>> fourSum(vector<int> &nums, int target) {
        int n = nums.size();
        vector<vector<int>> ans;
        sort(nums.begin(), nums.end());
        for(int i = 0;i<n;i++){
            if(i>0 && nums[i] == nums[i-1]) continue;
            for(int j = i+1;j<n;j++){
                if(j != i + 1 && nums[j] == nums[j-1]) continue;
                int k = j + 1;
                int l = n - 1;
                while(k < l){
                    long long sum = nums[i];
                    sum += nums[j];
                    sum += nums[k];
                    sum += nums[l];
                    if(sum == target){
                        vector<int> temp = {nums[i], nums[j], nums[k], nums[l]};
                        ans.push_back(temp);
                        k++;
                        l--;
                        while(k < l && nums[k] == nums[k-1]) k++;
                        while(k < l && nums[l] == nums[l-1]) l--;

                    } else if(sum < target) k++;
                    else l--;
                }
            }
        }
        return ans;
    }
```

C++ Autocomplete

```cpp
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        int n = nums.size();
        vector<vector<int>> ans;
        sort(nums.begin(), nums.end());
        for(int i = 0;i<n;i++){
            if(i>0 && nums[i] == nums[i-1]) continue;
            for(int j = i+1;j<n;j++){
                if(j != i + 1 && nums[j] == nums[j-1]) continue;
                int k = j + 1;
                int l = n - 1;
                while(k < l){
                    long long sum = nums[i];
                    sum += nums[j];
                    sum += nums[k];
                    sum += nums[l];
                    if(sum == target){
                        vector<int> temp = {nums[i], nums[j], nums[k], nums[l]};
                        ans.push_back(temp);
                        k++;
                        l--;
                        while(k < l && nums[k] == nums[k-1]) k++;
                        while(k < l && nums[l] == nums[l+1]) l--;
                    } else if(sum < target) {
                        k++;
                        while(k < l && nums[k] == nums[k-1]) k++;
                    } else {
                        l--;
                        while(k < l && nums[l] == nums[l+1]) l--;
                    }
                }
            }
        }
        // Remove duplicates
        sort(ans.begin(), ans.end());
        ans.erase(unique(ans.begin(), ans.end()), ans.end());
        return ans;
    }
};
```

T.C $\Rightarrow O(n^2 \times n) \approx O(n^3)$ S.C $\Rightarrow O(\text{no of quads})$