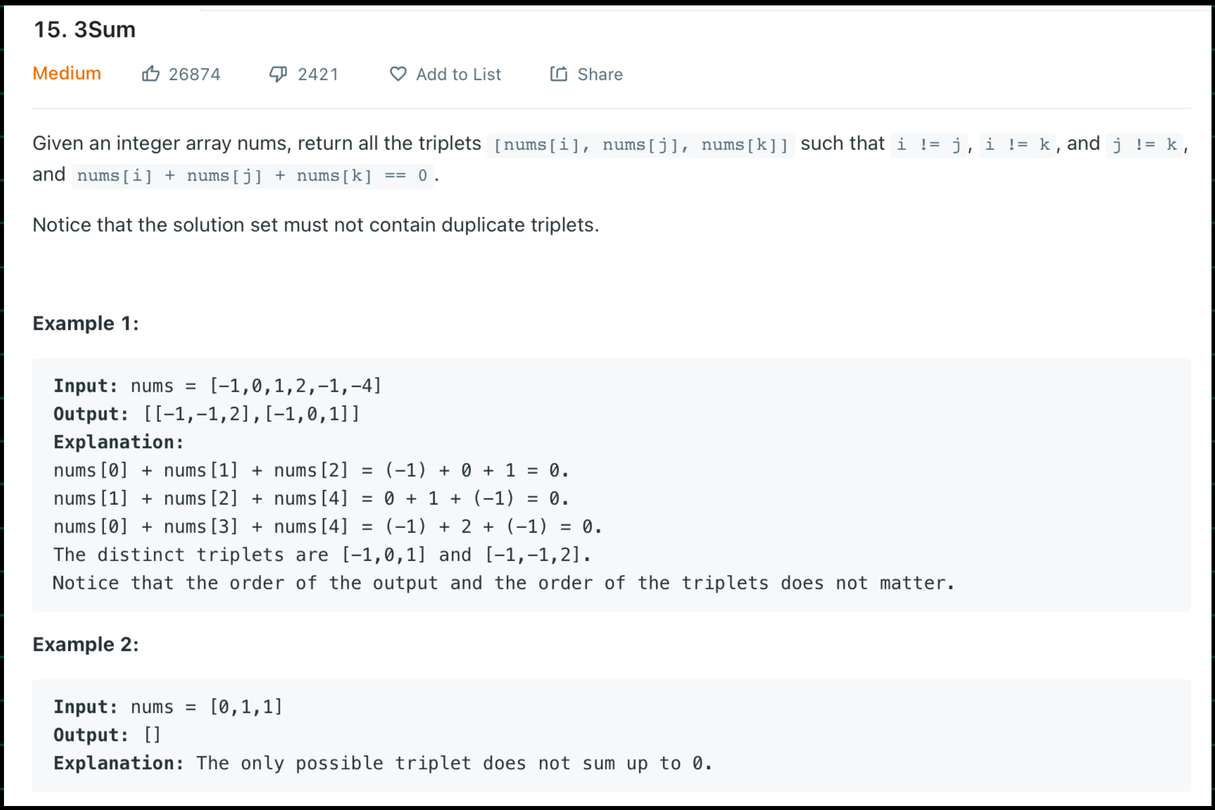
3 sum



15. 3Sum

Medium 26874 2421 Add to List Share

Given an integer array nums, return all the triplets `[nums[i], nums[j], nums[k]]` such that `i != j`, `i != k`, and `j != k`, and `nums[i] + nums[j] + nums[k] == 0`.

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]]
Explanation:
nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.
nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.
nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.
The distinct triplets are [-1,0,1] and [-1,-1,2].
Notice that the order of the output and the order of the triplets does not matter.
```

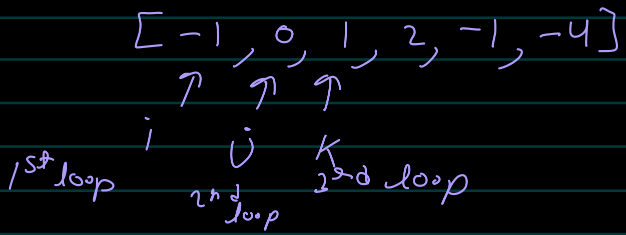
Example 2:

```
Input: nums = [0,1,1]
Output: []
Explanation: The only possible triplet does not sum up to 0.
```

Brute force :

Given Array = [-1, 0, 1, 2, -1, -4]

Choose element 3 element such that nums[i] + nums[j] + num[k] = k , i != j != k



```
loop(i : 0 -> n)
        loop(j : i -> n )
              loop ( k : j -> n)
                    if ( nums[i] + num[j] + num[k] == k)
                         store in vector [i, j, k]
                         sort( v) of vector
                         store in set (No duplicate)
```

}

}

}

store in vector<vector<int>> ans(st.begin - end)

T.C - $O(n^3)$

S.C - $O(3^* K)$ K is no. of triplets.

Better approch:

Similar to 2sum

nums[i] + nums[j] + nums[k] = 0

nums[i] + num[j] = - nums[k]

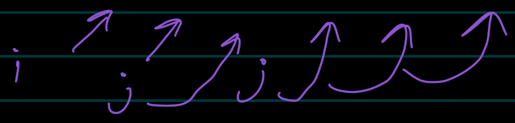
-(num[i] + num[j]) = num[k]

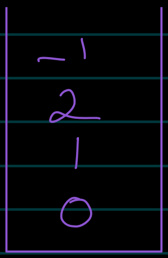
Run a loop for i and j

How to insert element in hashmap

arr[k] = -(arr[i] + arr[j])

[-1, 0, 1, 2, -1, -4]





- (-1 + 0)
= 1

Find (-1) in Set false

insert (j)
j++

- (-1 + 1)

= 1 - 1
= 0
Find (0) True

Insert(j)

Got triple

Sort (-1, 1, 0)

Store

$-(-1+2)$

$= 1-2$

$= -1$

Find (-1) False

insert (j)

$j{+}{+}$

$-(-1-1)$

$(1+1)$

$= 2$

Find (2) True

insert (j)

$j{+}{+}$

$-(-1-4)$

$1+4$

$= 5$

Find(5) False

$j<n$ False

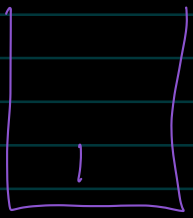
$i{+}{+}$

empty hashset

Found

sort$(-1, -1, 2)$

Store in Set



$[-1, 0, 1, 2, -1, -4]$

$\uparrow \quad \uparrow$

$i \quad j$

$-(0+1)$

$= -1$

insert(j)

j++

Go on till $i<n$

T.C - $O(n^2 \log n)$

S.C - $O(n)$

Optimize approch:-

Sort the given array and using 3 pointer

suppose we have array

[−2, −2, −2, −1, −1, −1, 0, 0, 0, 2, 2, 2, 2]

i ↗ j ↗ ↑ k

$i = 0$ Constant / will after first iteration ends

$j = i+1$ always / variable

$k = n-1$ always / variable

Using j and k find 3 elements which statisfy condition use Binary Search variation

```
while(j<k)
  sum = i+j+k   (add element)
  Check if it is greater than 0 then reduce k
  else check if smaller than 0 increase j
  otherwise we will have our triplet in Sorted order
        → v[i, j, k]
```

$i < j < k$ always since Sorted

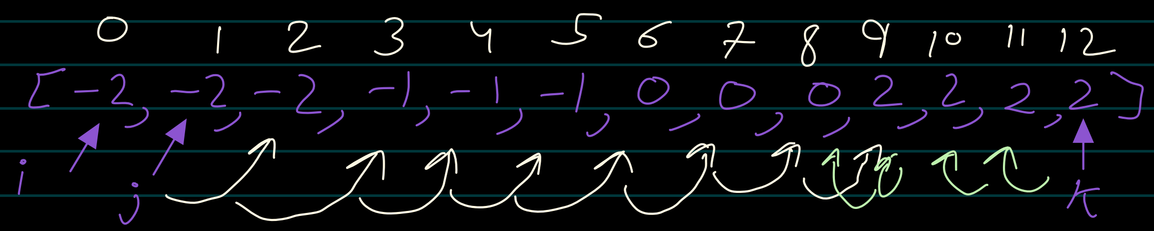
Now to avoid duplication we don't the element we inserted so

move k till num[k] == num[k+1]

move j till num[j] == num[j-1]

once we get all the triplets for i^{th} position

since we don't want find duplicate increase i till nums[i] == num[i-1]



i = 0 → j = 1 , k = 12

sum = -2 - 2 + 2
= -2 < 0

increase j++

Same j++

j = 3 , k = 12 , i = 0

sum = -2 - 1 + 2
= -1

increase j++
j++
j++

j = 6 , k = 12, i = 0

sum = 0 - 2 + 2
= 0

Found triplet

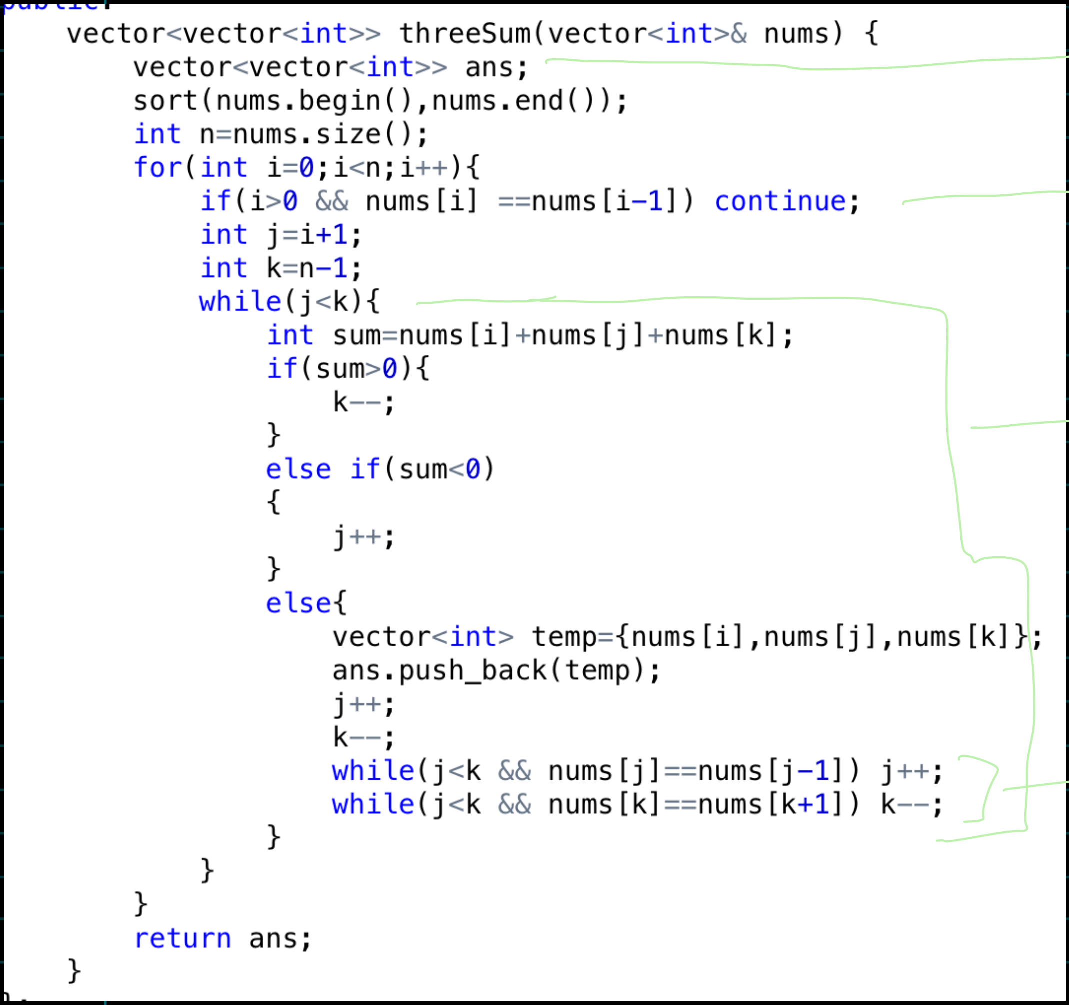
v(nums[i], nums[j], k)
ans.push_back(v)

while (j == 0) j++
while (k == 2) k--

while(j<k) False

i++
again j = i+1
k = n-1

Same process



```
    vector<vector<int>> threeSum(vector<int>& nums) {
        vector<vector<int>> ans;
        sort(nums.begin(),nums.end());
        int n=nums.size();
        for(int i=0;i<n;i++){
            if(i>0 && nums[i] ==nums[i-1]) continue;
            int j=i+1;
            int k=n-1;
            while(j<k){
                int sum=nums[i]+nums[j]+nums[k];
                if(sum>0){
                    k--;
                }
                else if(sum<0)
                {
                    j++;
                }
                else{
                    vector<int> temp={nums[i],nums[j],nums[k]};
                    ans.push_back(temp);
                    j++;
                    k--;
                    while(j<k && nums[j]==nums[j-1]) j++;
                    while(j<k && nums[k]==nums[k+1]) k--;
                }
            }
        }
        return ans;
    }
```

→ Ans stored

→ i!=0 first iteratin

→ Binary Search

→ Don't want Duplicate

$T.C \rightarrow O(n\log n) + O(n \times n)$

Sort

i loop

while

nested

$S.C - O(n)$